REQUETES SQL TOYS AND MODELS

```
USE toys_and_models;

### Ventes : Le nombre de produits vendus par catégorie et par mois,

### avec comparaison et taux de variation par rapport au même mois de l'année précédente.


WITH volume_categorie AS (

   -- Création de la CTE 'volume_ catégorie' pour stocker les commandes en volume par catégorie

SELECT  p.productline AS categorie,

YEAR(o.orderDate) AS Year,

MONTH(o.orderDate) AS Month,

sum(od.quantityOrdered) AS quantité_commandes,

LAG(sum(od.quantityOrdered), 1, 0) OVER (PARTITION BY p.productline, MONTH(o.orderDate)

-- Fonction LAG pour récuperer les quantités commandées pour le mois précédent, partitonner par productline

ORDER BY YEAR(o.orderDate)) AS precedente_quantité_commandes

FROM products p

JOIN orderdetails od ON p.productCode = od.productCode

JOIN orders o ON od.orderNumber = o.orderNumber

WHERE status != 'Cancelled'

-- Nous retirons les commandes annulées via la clause WHERE

GROUP BY categorie, Year, Month

ORDER BY categorie, Year, Month

 )

-- A partir de cette CTE,nous récupérons les Alias pour calculer le taux d'évolution arrondis à 2 décimales, pour chaques

-- pour chaques catgories, par an, par mois et par quantité commandée
```

```
SELECT categorie, Year, Month,
quantité_commandes,precedente_quantité_commandes,
ROUND(((quantité_commandes-
precedente_quantité_commandes)/precedente_quantité_commandes)*100,2) as
taux_evolution
FROM volume_categorie
ORDER BY categorie, Year DESC, Month DESC, taux_evolution DESC ;
```

FINANCES : Le chiffre d'affaires des commandes des deux derniers mois par pays.

```
SELECT Pays, chiffre_affaire, Years, Months, Rang
FROM
-- création d'une sous requête pour calculer le CA par pays, par an, par mois
-- et ordoner les résultats en créant un classement(RANK)
(SELECT c.country AS Pays,
SUM(od.quantityOrdered * od.priceEach) AS chiffre_affaire,
MONTH(o.orderDate) AS Months,
YEAR(o.orderDate) AS Years,
RANK() OVER(PARTITION BY c.country ORDER BY YEAR(o.orderDate),
MONTH(o.orderDate)) AS Rang,
-- la fonction RANK permet de créer un rang (classement) pour les résultats, ici les deux
derniers mois
o.status
FROM orderdetails od
JOIN orders o ON o.orderNumber = od.orderNumber
JOIN customers c ON o.customerNumber = c.customerNumber
GROUP BY c.country, Years, Months, o.status ) AS Rang_total
WHERE status != 'Cancelled' AND Rang <= 2
-- la clause WHERE nous permet de filtrer les résultats en retirant les commandes
annulée
```

```
-- et en sélectionnant les rang 1 et 2
ORDER BY Pays, Years, Months, Rang ASC;


### FINANCES : Les commandes qui n'ont pas encore été payées.


WITH
-- Etape 1:
-- Récupère les montants commandés par identifiants client et par commande
-- Exclu les commandes annulées
order_query AS (
  SELECT
    orders.customerNumber,
    orders.orderNumber,
    orders.status,
    SUM(orderdetails.quantityOrdered*orderdetails.priceEach) as ordered
  FROM orders
  JOIN orderdetails ON orders.orderNumber= orderdetails.orderNumber
  WHERE status != 'Cancelled'
  GROUP BY orders.orderNumber
  ),
-- Etape 2:
-- Récupère les montants payés par identifiants clients
payment_query AS (
  SELECT customerNumber,sum(amount) as total_paid
  FROM payments
  GROUP BY customerNumber
  )
-- combine la table des customers avec le tableau des paiements et des commandes
```

```
SELECT customers.customerName, total_paid,sum(ordered) as total_ordered,
    round(sum(ordered) - total_paid,2) as total_remain
FROM customers
JOIN payment_query ON payment_query.customerNumber = customers.customerNumber
JOIN order_query ON order_query.customerNumber = customers.customerNumber
GROUP BY (order_query.customerNumber)
HAVING round(sum(ordered) - total_paid,2) > 0
```

Logistique : Le stock des 5 produits les plus commandés.

```
WITH order_ranking AS (
    SELECT orderdetails.productCode,
        sum(orderdetails.quantityOrdered) as total_quantity_ordered,
        RANK() OVER (ORDER BY SUM(orderdetails.quantityOrdered) DESC) AS order_rank
    FROM orderdetails
    JOIN orders
     ON orders.orderNumber = orderdetails.orderNumber
    WHERE status != 'Cancelled'
    GROUP BY orderdetails.productCode
    ORDER BY total_quantity_ordered DESC
)
-- Etape 2 : Jointure avec la table produit pour :
-- Rapatrier le nom des 5 produits les plus commandés indépendamment des années
-- Rapatrier les quantités en stock
SELECT order_ranking.productCode, products.productName,order_rank,
```

```
    SUM(quantityInStock) AS total_quantity_in_stock
FROM order_ranking
JOIN products
 ON order_ranking.productCode = products.productCode
WHERE order_rank in (1,2,3,4,5)
GROUP BY order_ranking.productCode, products.productName,order_rank
ORDER BY order_rank
```

Ressources Humaines : Chaque mois, les 2 vendeurs ayant le plus haut chiffre d'affaires.

```
WITH ca_per_order AS (
 SELECT orders.customerNumber,
    orders.orderNumber,
    year(orderDate) as year_date,
    month(orderDate) as month_date,
    sum(orderdetails.quantityOrdered*orderdetails.priceEach) as sale_revenu
 FROM orders
 JOIN orderdetails
  ON orders.orderNumber= orderdetails.orderNumber
 WHERE status != 'Cancelled'
 GROUP BY orders.customerNumber,orders.orderNumber,month_date,year_date
),
-- Etape 2 :
-- Lier table customer pour trouver le vendeur en charge du client à l'origine de la commande
ca_per_saler AS (
  SELECT
    year_date,
```

```
    month_date,
    customers.salesRepEmployeeNumber,
    SUM(sale_revenu) as total_sale_revenu
 FROM ca_per_order
 JOIN customers
  ON customers.customerNumber= ca_per_order.customerNumber
 GROUP BY customers.salesRepEmployeeNumber,month_date,year_date
),
-- Etape 3 :
-- Créer un rang des vendeurs
-- Classer les données dans chaque partition par année et mois,
-- par ordre décroissant pour obtenir le rang 1 = vendeur avec le plus de CA
saler_rank AS (
  SELECT salesRepEmployeeNumber,
    year_date,
    month_date,
    total_sale_revenu,
    lastName,
    firstName,
    jobTitle,
    RANK() OVER (PARTITION BY year_date,month_date ORDER BY total_sale_revenu
DESC) AS sales_rank
  FROM ca_per_saler
  JOIN employees
   ON ca_per_saler.salesRepEmployeeNumber = employees.employeeNumber
)
-- Etape 4 :
-- Résultat de la requête
```

```
-- En filtrant sur les deux vendeurs ayant le plus haut CA par mois
SELECT year_date,
    month_date,
    lastName,
    firstName,
    jobTitle,
    total_sale_revenu
FROM saler_rank
WHERE sales_rank IN (1,2)
```