Contents

JOUR 10M

Télécharge les fichiers dont tu as besoin

Tu trouveras ici le fichier à télécharger pour le chapitre :

Le livecode_DML_int.SQL contient toutes les requêtes que l'instructeur va écrire dans ce chapitre, afin que tu puisses y avoir accès facilement pour les tester en même temps que l'instructeur, les modifier etc..

La base de données: orders.csv

- **Livecode_SQL_avance.sql**5,8 KO

TÉLÉCHARGEMENTS'OUVRE DANS UNE NOUVELLE FENÊTRE

orders.csv518 OCTETS

Window functions

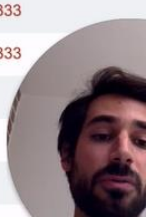
Window function

Dans votre magasin, vous souhaitez savoir quel est le panier moyen.
→ Vous savez utiliser une function agrégative (AVG)

avg(order_amount)
10050

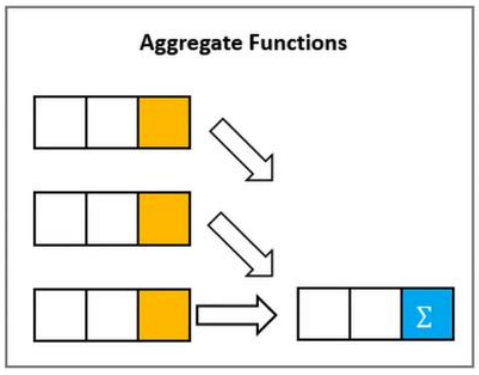
Cependant, imaginons que vous ayez de nombreux magasins dans différentes villes. Vos voulez afficher le panier moyen par ville **pour chaque commande**

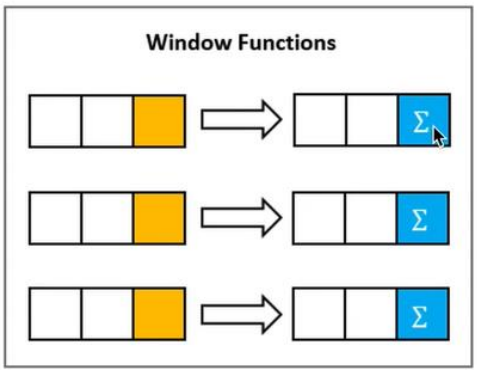
city	order_amount	average_order_amount
Arlington	15000	12333.3333
Arlington	20000	12333.3333
Arlington	2000	12333.3333
GuildFord	10000	12625
GuildFord	15000	12625
GuildFord	25000	12625



Window function

Contrairement aux fonctions d'agrégation usuelles, l'usage d'une window function ne groupe pas les lignes lors de l'affichage du résultat de la requête





MariaDB | Copie de blabla_data_exo_update...

```sql
SELECT ride_id, starting_city_id, contribution_per_passenger,
    AVG(contribution_per_passenger) over(PARTITION BY starting_city_id)
                                        AS start_city_avg_contribution
    FROM rides r ;
```

Window functions - Examples dont Rank et Row_number

```sql
SELECT*
FROM orders_dataset
```

order_id	order_date	customer_name	city	order_amount
1001	2017-02-02	David Smith	GuildFord	10000
1002	2017-02-03	David Jones	Arlington	20000
1003	2017-02-04	John Smith	Shalford	5000
1004	2017-02-05	Michael Smith	GuildFord	15000
1005	2017-02-06	David Williams	Shalford	7000

Window function (2)

Faire des calculs à partir d'une table

```sql
SELECT order_id,order_date, customer_name, city, order_amount, SUM(order_amount)
over(PARTITION BY city) AS grand_total
FROM orders
```

```
4
5 # quelle est les chifre d'affer total des villes
6 SELECT *, SUM(order_amount)
7 over(PARTITION BY city) AS tot_amnt_per_city
8 FROM orders
9
```

order_id	order_date	customer_name	city	order_amount	tot_amnt_per_city
1002	2017-02-03	David Jones	Arlington	20000	37000
1008	2017-02-09	David Brown	Arlington	2000	37000
1007	2017-02-08	Andrew Smith	Arlington	15000	37000
1006	2017-02-07	Paum Smith	GuildFord	25000	50500

Window function (3)

Example 2, with average calculation

```
SELECT order_id, order_date, customer_name, city, order_amount, AVG(order_amount)
OVER(PARTITION BY city, MONTH(order_date)) AS   average_order_amount
FROM Orders
```

```
11
12 SELECT order_id, order_date, customer_name,city, order_amount, AVG(order_amount)
13 over(PARTITION BY city, month(order_date)) AS avr_order_amnt
14 FROM orders_dataset
15
```

order_id	order_date	customer_name	city	order_amount	avr_order_amnt
1007	2017-02-08	Andrew Smith	Arlington	15000	12333.3333
1002	2017-02-03	David Jones	Arlington	20000	12333.3333
1008	2017-02-09	David Brown	Arlington	2000	12333.3333
1004	2017-02-05	Michael Smith	GuildFord	15000	12625.0000

Window function (4)

Showing the number of rows

```
SELECT order_id,order_date,customer_name,city, order_amount,
ROW_NUMBER() OVER(ORDER BY order_id) row_number
FROM Orders;
```

```
1 SELECT order_id, order_date, customer_name,
2 row_number() over(PARTITION BY order_date ORDER BY order_date) AS chronu
3 FROM orders_dataset
```

order_id	order_date	customer_name	chronu
1001	2017-02-02	David Smith	1
1002	2017-02-03	David Jones	1
1003	2017-02-04	John Smith	1

Window function (5)

Showing max by category

```
SELECT order_id, order_date, customer_name, city, order_amount
 ,MAX(order_amount) OVER(PARTITION BY city) AS maximum_order_amount
FROM Orders;
```

Utiliser Lead et Lag pour comparer les valeurs consécutives

Window function (7)

Combining PARTITION BY and ORDER BY

```
SELECT c.city_name, t.*
FROM
(
SELECT ride_id, starting_city_id, contribution_per_passenger,
        ROW_NUMBER() OVER(PARTITION BY starting_city_id
                          ORDER BY contribution_per_passenger) AS row_number_price
FROM rides r
) t
INNER JOIN cities c ON c.city_id = t.starting_city_id
WHERE row_number_price <= 30;
```

MariaDB | Copie de blabla_data_exo_update... | mariadb.21 | mariadb.24

```
1 SELECT ride_id, starting_city_id, contribution_per_passenger,
2        ROW_NUMBER() OVER(PARTITION BY starting_city_id
3                          ORDER BY contribution_per_passenger) AS row_number_price
4 FROM rides r
```

ride_id	starting_city_id	contribution_per_passenger	row_number_price
211	1	5	1
51	1	5	2
358	1	5	3
348	1	5	4

```sql
SELECT c.city_name, t.*
FROM
(
SELECT ride_id, starting_city_id, contribution_per_passenger,
        ROW_NUMBER() OVER(PARTITION BY starting_city_id
                          ORDER BY contribution_per_passenger) AS row_number_price
FROM rides r
) t
INNER JOIN cities c ON c.city_id = t.starting_city_id
WHERE row_number_price <= 30;
```

city_name	ride_id	starting_city_id	contribution_per_passenger	row_number_price
Paris	70	1	15	30
Paris	53	1	15	29
Paris	371	1	15	28

Window function (8) - Lead & Lag

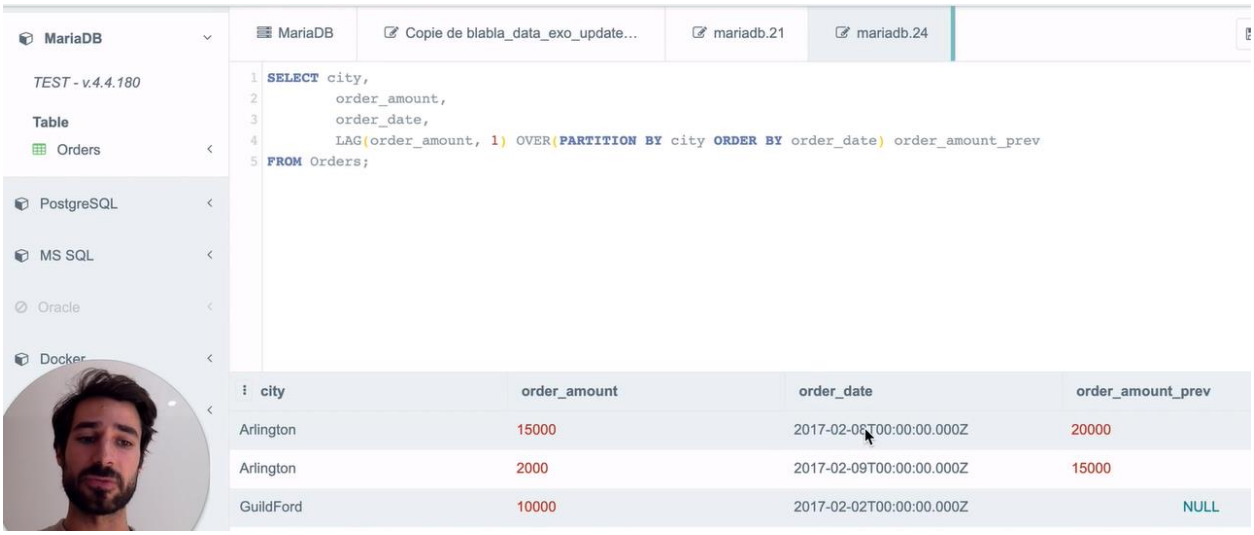
```sql
SELECT order_id,customer_name,city, order_amount,order_date,
LAG(order_date,1) OVER(ORDER BY order_date) prev_order_date
FROM Orders
```

```sql
SELECT order_id,customer_name,city, order_amount,order_date,
LEAD(order_date,1) OVER(ORDER BY order_date) prev_order_date
FROM Orders
```

```
SELECT order_id,customer_name,city, order_amount,order_date,
LAG(order_date,1) OVER(ORDER BY order_date) prev_order_date
FROM Orders
```

order_id	customer_na...	city	order_amount	order_date	prev_order_date
1001	David Smith	GuildFord	10000	2017-02-02T0...	NULL
1002	David Jones	Arlington	20000	2017-02-03T0...	2017-02-02T00:00:0...
1003	John Smith	Shalford	5000	2017-02-04T0...	2017-02-03T00:00:0...
1004	Michael Smith	GuildFord	15000	2017-02-05T0...	2017-02-04T00:00:0...
1005	David Williams	Shalford	7000	2017-02-06T0...	2017-02-05T00:00:0...

Exercice d'application 2



MariaDB | Copie de blabla_data_exo_update… | mariadb.21 | mariadb.24

```
SELECT city,
       order_amount,
       order_date,
       (order_amount / LAG(order_amount, 1) OVER(PARTITION BY city ORDER BY order_date) - 1)*100 AS percentage
FROM Orders;
```

city	order_amount	order_date	percentage
Arlington	20000	2017-02-03T00:00:00.000Z	NULL
Arlington	15000	2017-02-08T00:00:00.000Z	-25
Arlington	2000	2017-02-09T00:00:00.000Z	-86.6667

MariaDB | Copie de blabla_data_exo_update… | mariadb.21 | mariadb.24

```
WITH avg_evolution_percentage AS
     (
     SELECT city,
            order_amount,
            order_date,
            (order_amount / LAG(order_amount, 1) OVER(PARTITION BY city ORDER BY order_date) - 1)*100 AS percentage
     FROM Orders
     )
SELECT city, AVG(percentage) AS avg_percentage
FROM avg_evolution_percentage
GROUP BY city;
```

city	avg_percentage
Arlington	-55.83335
GuildFord	6.22223333
Shalford	-22.85715

Exercice d'application 3

```sql
WITH
    price_type AS
        (
        SELECT ride_id,
                starting_city_id,
                contribution_per_passenger,
                AVG(contribution_per_passenger) OVER(PARTITION BY starting_city_id) city_avg_contrib
        FROM rides
        ),
    contrib_algorithm AS
        (
        SELECT ride_id,
                starting_city_id,
                contribution_per_passenger,
                CASE
```

ride_id	starting_city_id	contribution_per_passenger	price_indication
221	1	15	aubaine
322	1	5	aubaine
402	1	5	aubaine

```sql
        ),
    contrib_algorithm AS
        (
        SELECT ride_id,
                starting_city_id,
                contribution_per_passenger,
                CASE
                    WHEN contribution_per_passenger > city_avg_contrib THEN 'arnaque'
                    ELSE 'aubaine'
                END AS price_indication
        FROM price_type
        )
SELECT *
FROM contrib_algorithm
WHERE price_indication = 'aubaine';
```

ride_id	starting_city_id	contribution_per_passenger	price_indication
221	1	15	aubaine
322	1	5	aubaine
402	1	5	aubaine

JOUR 10A

Tu trouveras ici un pdf synthétisant toutes les questions à résoudre pour bien maitriser les combinaisons de données SQL.

- **Question_J7_exos.docx**14 KO

TÉLÉCHARGEMENTS'OUVRE DANS UNE NOUVELLE FENÊTRE

Tu peux trouver ici un fichier SQL qui contient les corrigés détaillés des questions, les vidéos ci-après sont construites autour de ce corrigé où les méthodes sont détaillées, on te conseille de bien prendre le temps avant de regarder les corrigés, n'oublie pas que le forum est le lieu où coopérer pour avancer sur les exercices avant de consulter les corrigés détaillés !

Une fois que tu as consulté les corrigés, sens-toi libre de participer au forum et d'aiguiller les personnes posant des questions en leur donnant des indices, c'est un très bon moyen d'apprentissage et une preuve de sa compréhension lorsqu'on parvient à aider ses pairs !