

Contents

JOUR 8M	1
Une autre approche conditionnelle : Le CASE WHEN.....	1
Gérer les dates avec SQL.....	2
Faciliter ses requêtes complexes avec Les Common Table Expressions	3
Cours en ligne :.....	6
JOUR 9.....	8
Questions 1 - 3	8
Question 4.....	10
Question 1 - 2 Intermédiaire.....	10
Questions 5 - 6 Intermédiaire	15
Question 7 intermédiaire.....	17

JOUR 8M

🔖 Objectifs

- Créer une colonne utilisant des conditions si... sinon... (ex : je crée une colonne tranche_age : si $18 < \text{age} \leq 30$, alors la cellule prend la valeur "18-29", si $30 < \text{age} \leq 40$ alors la cellule prend la valeur "31-40" etc.)
- Manipuler les dates pour faire des séries temporelles
- Optimiser tes requêtes difficiles avec les common table expressions (CTE)

Ici, les leçons en Bonus sont optionnelles, on te conseille de les suivre uniquement si tu te sens à l'aise dur SQL et que tu as du temps ! De même pour les exercices Bonus

Télécharge les fichiers dont tu as besoin

[Livecode_SQL_avance-J8](#)

Une autre approche conditionnelle : Le CASE WHEN

La commande CASE permet de catégoriser des données. Voyez-la comme une nouvelle colonne que vous appelez dans le SELECT

- Syntaxe : CASE WHEN condition THEN resultat_1 ELSE resultat_2 END AS nom_colonne_case
- En cas de sélection de plusieurs colonnes dans votre requête (y compris avec *), le CASE doit TOUJOURS être précédé d'une virgule
- On peut utiliser un CASE avec une autre fonction d'agrégation, comme un COUNT
- Dans MySQL ou PostgreSQL, on peut utiliser l'alias du CASE dans un group by

Récap - notions clés

Les différentes approches de la combinaison de données



La commande CASE

Les commandes CASE permettent de catégoriser les données

```
SELECT *,
CASE WHEN lt.type = 'petit' THEN 'gratuit'
      ELSE 'payant'
      END AS price
FROM luggage_types lt;
```

Gérer les dates avec SQL

Il existe 4 formats de date : DATE, DATETIME, TIMESTAMP, TIME.

- DATE : format **YYYY-MM-DD**
- DATETIME & TIMESTAMP : format **YYYY-MM-DD hh:mm:ss** (principale différence : la prise en compte du fuseau horaire avec TIMESTAMP)
- TIME : format **hh:mm:ss**

Beaucoup de fonction tournent autour de l'utilisation des dates. Par exemple :

- NOW() : donne la date d'aujourd'hui au format datetime
- DATE() : permet de passer d'un format DATETIME à un format DATE
- DATEDIFF() : renvoie un nombre de jours entre les 2 dates entrées en paramètre
- DATEADD() : ajoute un intervalle de temps à partir d'une date entrée en paramètre
- DATESUB() : soustrait un intervalle de temps à partir d'une date entrée en paramètre

Date – Les types principaux

Date

- YYYY-MM-DD
- Exemple : 2018-12-30

Datetime / Timestamp

- YYYY-MM-DD HH:MM:SS
- Exemple : 2018-12-30 13:10:04

Time

- HH:MM:SS
- Exemple : 13:10:04

Date (2)

Exemples:

```

SELECT now(); -- Timestamp
SELECT DATE(now()); -- Date
SELECT DATE(now()) > '2017-12-31';
SELECT now() > '2017-12-31';

SELECT DATEDIFF(DATE(now()), DATE("2017-06-15"));
SELECT DATE_ADD("2017-06-15", INTERVAL 10 YEAR);

```

Faciliter ses requêtes complexes avec Les Common Table Expressions

Il existe différents types de CTE : VIEW, WITH et TEMPORARY TABLE

VIEW :

- Permet de mémoriser une table virtuelle. Cette table n'est pas intégrée au schéma, et ne prend aucun espace de stockage : la requête sera soumise à chaque fois que vous appellerez cette table virtuelle
- Syntaxe : CREATE VIEW nom_vue AS...

TEMPORARY TABLE :

- Propre à une session : chaque utilisateur peut avoir sa temporary table. Aussi, la temporary table disparaît à la fin de la session

WITH :

- Permet de réorganiser une longue requête, en créant des tables intermédiaires au début de la requête. Très utile lorsque l'on veut faire des jointures multiples avec des tables comportant des conditions
- Syntaxe : WITH nom_table_intermediaire AS (...)

Permanent table**CTE : Common Table Expression**

En SQL, une vue est une table virtuelle construite par le résultat d'une requête SQL

Une vue contient lignes et colonnes, comme une véritable table

```
CREATE VIEW nom_vue AS
SELECT colonne1, colonne2
FROM nom_table
WHERE conditions;
```



```

1 CREATE VIEW table_age_category AS
2 SELECT
3     CASE WHEN DATEDIFF(NOW(), birthdate) / 365 < 25 THEN '18-24'
4     WHEN DATEDIFF(NOW(), birthdate) / 365 BETWEEN 25 AND 39 THEN '25-39'
5     WHEN DATEDIFF(NOW(), birthdate) / 365 BETWEEN 40 AND 60 THEN '40-60'
6     ELSE '60+'
7     END AS age_category,
8     COUNT(*) AS nb_members,
9     ROUND(COUNT(*)*100 / (SELECT COUNT(*) FROM members), 2) AS percentage
10 FROM members
11 GROUP BY age_category;

```

Temporary table

WITH

Que faire si on ne veut pas créer un objet permanent ?

Il est possible de créer une requête complexe étape par étape

→ Il suffit de déclarer des tables intermédiaires qui seront utilisés plus tard dans la requête

```

WITH pet_lovers AS
(
    SELECT *
    FROM members m
    WHERE pet_preference = 'yes'
)
SELECT member_id, email, birthdate
FROM pet_lovers
WHERE birthdate > DATE('1990-01-01')

```

```

1 WITH pet_lovers AS
2 (
3     SELECT *
4     FROM members m
5     WHERE pet_preference = 'yes'
6 )
7 SELECT member_id,email, birthdate
8 FROM pet_lovers
9 WHERE birthdate > DATE('1990-01-01');

```

member_id	email	birthdate
mem-1003508	Blanch_Wertman73@mail.com	1991-07-15T00:00:00.000Z
mem-1167206	Marcy.Escobar38@HEC.com	1997-04-20T00:00:00.000Z
mem-1599896	Jaleesa_Iraheta51@hotmail.com	2000-01-09T00:00:00.000Z

```

1 WITH
2     pet_lovers AS
3     (
4         SELECT *
5         FROM members m
6         WHERE pet_preference = 'yes'
7     ),
8     pet_haters AS
9     (
10        SELECT *
11        FROM members m

```

Cours en ligne :

JOUR 9

Questions 1 - 3

Afficher une nouvelle colonne 'autorisé à Paris' qui prend la valeur 'oui' ou 'non' selon si la voiture en question sera autorisée à Paris en 2022.

En effet, les voitures fabriquées avant 2004 seront interdites à cette date.

```

1 SELECT *,
2     CASE
3         WHEN cars.year < 2004 THEN 'Non'
4         ELSE 'Oui'
5     END AS droit_circu_paris
6 FROM cars;

```

i	car_id	maker	CO2_code	colour	year	plate	droit_circu_f
1		BMW	653Z	red	1994	18-FIR-87	Non
10		Land Rover	321E	red	2007	42-KOD-73	Oui
11		Fiat	332E	white	1993	39-KAE-82	Non
12		Chrysler	111A	gray	2001	47-IHL-13	Non

Afficher les fabricants qui ont plus de 3 voitures enregistrées dans la base.

MariaDB	Copie de blabla_data_exo_update...	mariadb.21	mariadb.21
---------	------------------------------------	------------	------------

```

1 SELECT maker, COUNT(*) AS nb_cars
2 FROM cars
3 GROUP BY maker
4 HAVING nb_cars > 3;

```

maker	nb_cars
Land Rover	5

Calculer le nombre de courses faites par chaque voiture et la contribution totale perçue. Afficher les 10 premiers résultats (choisir l'une des deux métriques)

```

1 SELECT mc.car_id, COUNT(r.ride_id) AS nb_trips, SUM(contribution_per_passenger * number_seats) AS total_contrib
2 FROM rides r
3 INNER JOIN member_car mc
4     ON r.member_car_id = mc.member_car_id
5 GROUP BY mc.car_id
6 ORDER BY total_contrib DESC
7 LIMIT 10;

```

car_id	nb_trips	total_contrib
8	49	1525
4	52	1500
6	48	1465

Question 4

Créer une nouvelle catégorie dans cities qui indique si la cité est dans le nord ou le sud (vous choisissez quand séparer).

```

1 SELECT *,
2     CASE
3         WHEN city_name IN ('Lyon', 'Marseille') THEN 'South'
4         ELSE 'North'
5     END AS city_position
6 FROM cities;

```

city_id	city_name	state	country	city_position
1	Paris	IDF	France	North
2	Versailles	Yvelines	France	North
3	Tour	Indre-et-Loire	France	North

Question 1 - 2 Intermédiaire

Reprendre les résultats de la question 4 pour afficher la répartition des courses selon leurs positions nord/sud.

Pour cela, afficher le nombre de courses de Nord au Nord, du Nord au Sud, du Sud au Nord, du Sud au Sud.

```

1 WITH table_city_position AS
2   (
3     SELECT city_id, city_name,
4           CASE
5             WHEN city_name IN ('Lyon', 'Marseille') THEN 'South'
6             ELSE 'North'
7           END AS city_position
8     FROM cities
9   )
10 SELECT *
11 FROM rides r
12 JOIN table_city_position AS tcp_dep
13   ON r.starting_city_id = tcp_dep.city_id
14 JOIN table_city_position AS tcp_arr

```

```

1 WITH table_city_position AS
2   (
3     SELECT city_id, city_name,
4           CASE
5             WHEN city_name IN ('Lyon', 'Marseille') THEN 'South'
6             ELSE 'North'
7           END AS city_position
8     FROM cities
9   )
10 SELECT tcp_dep.city_position AS position_dep, tcp_arr.city_position AS position_arr, COUNT(*) AS nb_trips
11 FROM rides r
12 JOIN table_city_position AS tcp_dep
13   ON r.starting_city_id = tcp_dep.city_id
14 JOIN table_city_position AS tcp_arr
15   ON r.destination_city_id = tcp_arr.city_id
16 GROUP BY tcp_dep.city_position, tcp_arr.city_position;

```

position_dep	position_arr	nb_trips
North	North	226
North	South	126
South	North	130

Trouver combien de courses au départ du nord ont été prises par chaque membre.

```

1 SELECT member_id, COUNT(*) AS nb_trips
2 FROM rides r
3 JOIN city_position cp
4     ON r.starting_city_id = cp.city_id
5 JOIN member_car mc
6     ON r.member_car_id = mc.member_car_id
7 WHERE cp.city_position = 'North'
8 GROUP BY member_id;

```

member_id	nb_trips
mem-1353047	6
mem-1391067	18
mem-1462013	17

```

1 CREATE VIEW city_position AS
2 SELECT city_id, city_name,
3         CASE
4             WHEN city_name IN ('Lyon', 'Marseille') THEN 'South'
5             ELSE 'North'
6         END AS city_position
7 FROM cities;

```

```

1 SELECT *
2 FROM rides r
3 JOIN city_position cp
4     ON r.starting_city_id = cp.city_id
5 JOIN member_car mc
6     ON r.member_car_id = mc.member_car_id
7 LIMIT 10;

```

```
< 1 SELECT member_id, COUNT(*) AS nb_trips
< 2 FROM rides r
< 3 JOIN city_position cp
< 4     ON r.starting_city_id = cp.city_id
< 5 JOIN member_car mc
< 6     ON r.member_car_id = mc.member_car_id
< 7 WHERE cp.city_position = 'North'
< 8 GROUP BY member_id
< 9 ORDER BY nb_trips DESC;| I
```

member_id	nb_trips
mem-2426507	21
mem-4344480	18
mem-3122499	18

Créer de nouvelles colonnes dans rides pour afficher le jour de la semaine, le numéro du mois et de l'année de la course.

```

1 SELECT ride_id,
2        DAYOFWEEK(departure_date) AS departure_day,
3        MONTH(departure_date) AS departure_month,
4        YEAR(departure_date) AS departure_year,
5        CASE
6            WHEN DAYOFWEEK(departure_date) = 1 THEN 'Sunday'
7            WHEN DAYOFWEEK(departure_date) = 2 THEN 'Monday'
8            WHEN DAYOFWEEK(departure_date) = 3 THEN 'Tuesday'
9            WHEN DAYOFWEEK(departure_date) = 4 THEN 'Wednesday'
10           WHEN DAYOFWEEK(departure_date) = 5 THEN 'Thursday'
11           WHEN DAYOFWEEK(departure_date) = 6 THEN 'Friday'
12           WHEN DAYOFWEEK(departure_date) = 7 THEN 'Saturday'
13        END AS day_name
14 FROM rides;

```

ride_id	departure_day	departure_month	departure_year	day_name
1	2	3	2020	Monday
10	4	4	2020	Wednesday
100	5	1	2020	Thursday

Retrouver toutes les courses qui ont eu lieu un lundi.

```

1 SELECT ride_id,
2        DAYOFWEEK(departure_date) AS departure_day,
3        CASE
4            WHEN DAYOFWEEK(departure_date) = 1 THEN 'Sunday'
5            WHEN DAYOFWEEK(departure_date) = 2 THEN 'Monday'
6            WHEN DAYOFWEEK(departure_date) = 3 THEN 'Tuesday'
7            WHEN DAYOFWEEK(departure_date) = 4 THEN 'Wednesday'
8            WHEN DAYOFWEEK(departure_date) = 5 THEN 'Thursday'
9            WHEN DAYOFWEEK(departure_date) = 6 THEN 'Friday'
10           WHEN DAYOFWEEK(departure_date) = 7 THEN 'Saturday'
11        END AS day_name
12 FROM rides
13 HAVING day_name = 'Monday';

```

ride_id	departure_day	day_name
1	2	Monday
112	2	Monday
120	2	Monday

Questions 5 - 6 Intermédiaire

Quelle part des courses ont été faites par des conducteurs de plus de 25 ans ?

```

1 SELECT COUNT(*)*100 / (SELECT COUNT(*) FROM rides) AS ratio_over_25
2 FROM rides r
3 JOIN member_car mc
4   ON r.member_car_id = mc.member_car_id
5 JOIN members m
6   ON mc.member_id = m.member_id
7 WHERE m.birthdate < DATE_SUB(NOW(), INTERVAL 25 YEAR);

```

ratio_over_25

82.4

2eme code

```

1 SELECT COUNT(over_25.ride_id) / (SELECT COUNT(r3.ride_id) FROM rides r3)*100 -- Notez que la deuxième sous-requête est
2 FROM (SELECT r2.ride_id, m.member_id, (YEAR(NOW()) - YEAR(m.birthdate)) AS DIF, YEAR(m.birthdate), YEAR(NOW()) -
3 FROM rides r2
4 LEFT JOIN member_car mc ON r2.member_car_id = mc.member_car_id
5 LEFT JOIN `members` m ON mc.member_ID = m.member_id
6 HAVING DIF>25) AS over_25;

```

ratio_over_25

82.4



Créer une vue de la question précédente.

```
1 CREATE VIEW OVER_25 AS
2 SELECT COUNT(*) *100 / (SELECT COUNT(*) FROM rides) AS ratio_over_25
3 FROM rides r
4 JOIN member_car mc
5     ON r.member_car_id = mc.member_car_id
6 JOIN members m
7     ON mc.member_id = m.member_id
8 WHERE m.birthdate < date_sub(now(), INTERVAL 25 YEAR);
```

```
1 SELECT *
2 FROM OVER_25;
```

ratio_over_25

82.4

Question 7 intermédiaire

Reprendre la question avec un WITH :
afficher la répartition des courses selon leurs positions nord/sud.
Pour cela, afficher le nombre de courses de Nord au Nord, du Nord au Sud, du Sud au Nord, du Sud au Sud.

```
1 WITH table_city_position AS
2   (
3     SELECT city_id, city_name,
4           CASE
5             WHEN city_name IN ('Lyon', 'Marseille') THEN 'South'
6             ELSE 'North'
7           END AS city_position
8     FROM cities
9   )
10 SELECT tcp_dep.city_position AS position_dep, tcp_arr.city_position AS position_arr, COUNT(*) AS nb_trips
11 FROM rides r
12 JOIN table_city_position AS tcp_dep
13   ON r.starting_city_id = tcp_dep.city_id
14 JOIN table_city_position AS tcp_arr
15   ON r.destination_city_id = tcp_arr.city_id
16 GROUP BY position_dep, position_arr;
```

position_dep	position_arr	nb_trips
North	North	226
North	South	126

```

1 SELECT `position_depart`, `position_arrivee`, COUNT(r.ride_id) AS total_ride
2 FROM rides AS r
3 JOIN (SELECT city_id, city_name,
4       CASE
5         WHEN city_name LIKE 'Lyon' OR city_name LIKE 'Marseille' THEN 'South'
6         ELSE 'North'
7       END AS `position_depart`
8 FROM cities AS cp) AS c
9   ON r.starting_city_id = c.city_id
10
11 JOIN (SELECT city_id, city_name,
12       CASE
13         WHEN city_name LIKE 'Lyon' OR city_name LIKE 'Marseille' THEN 'South'
14         ELSE 'North'
15       END AS `position_arrivee`
16 FROM cities AS cp2) AS c2
17   ON r.destination_city_id = c2.city_id

```

```

3 JOIN (SELECT city_id, city_name,
4       CASE
5         WHEN city_name LIKE 'Lyon' OR city_name LIKE 'Marseille' THEN 'South'
6         ELSE 'North'
7       END AS `position_depart`
8 FROM cities AS cp) AS c
9   ON r.starting_city_id = c.city_id
10
11 JOIN (SELECT city_id, city_name,
12       CASE
13         WHEN city_name LIKE 'Lyon' OR city_name LIKE 'Marseille' THEN 'South'
14         ELSE 'North'
15       END AS `position_arrivee`
16 FROM cities AS cp2) AS c2
17   ON r.destination_city_id = c2.city_id
18 GROUP BY `position_depart`, `position_arrivee`

```

position_dep	position_arr	nb_trips
North	North	226
North	South	126