

# TP - MySQL

**FERTILLE Eliott**

03/11/2023

—

Conception base de données

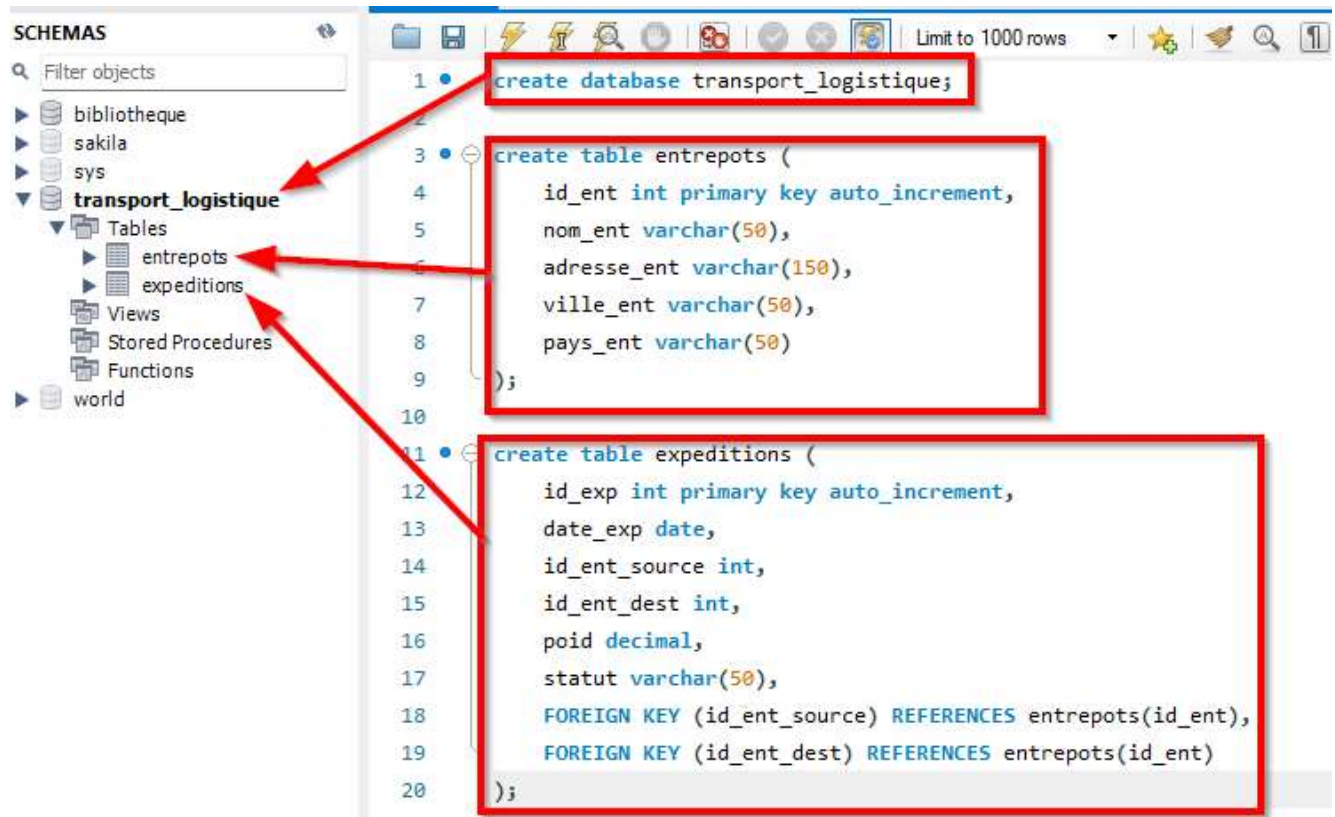
—

Mr. KAKE

---

# Création de la base de données

Pour créer une base de données avec deux tables « entrepots » et « expéditions », j'ai utilisé les requêtes CREATE DATABASE et CREATE TABLE :



The screenshot shows a MySQL database management tool interface. On the left, the 'SCHEMAS' panel displays a tree view of databases: 'bibliotheque', 'sakila', 'sys', 'transport\_logistique', 'Views', 'Stored Procedures', 'Functions', and 'world'. The 'transport\_logistique' database is selected and expanded, showing its 'Tables' (entrepots, expéditions), 'Views', 'Stored Procedures', and 'Functions'. On the right, the SQL editor shows the following code:

```
1 • create database transport_logistique;
2
3 • create table entrepots (
4     id_ent int primary key auto_increment,
5     nom_ent varchar(50),
6     adresse_ent varchar(150),
7     ville_ent varchar(50),
8     pays_ent varchar(50)
9 );
10
11 • create table expéditions (
12     id_exp int primary key auto_increment,
13     date_exp date,
14     id_ent_source int,
15     id_ent_dest int,
16     poid decimal,
17     statut varchar(50),
18     FOREIGN KEY (id_ent_source) REFERENCES entrepots(id_ent),
19     FOREIGN KEY (id_ent_dest) REFERENCES entrepots(id_ent)
20 );
```

Red arrows point from the 'entrepots' and 'expéditions' tables in the left panel to their respective CREATE TABLE statements in the SQL editor. Another red arrow points from the 'transport\_logistique' database in the left panel to the CREATE DATABASE statement in the SQL editor.

Nous avons maintenant une base de données MySQL nommé « transport\_logistique » qui contient les tables « entrepots », composée d'un id, d'un nom d'entrepôt, d'une adresse, d'une ville et d'un pays pour le moment.

Et de « expéditions » composée d'un id, d'une date d'expédition, d'un id entrepôts de source lié à l'id de entrepôts, d'un id entrepôts de destination lié à l'id de entrepôts, d'un poid et d'un statut.

# Jeu de données

Pour ajouter des données de test dans ma base, j'utilise une requête INSERT avec des données exemple en VALUES :

```
1  insert into entrepots(nom_ent, adresse_ent, ville_ent, pays_ent)
2  values ("Lyon Est", "2 Rue Jean Jaures", "Lyon", "France");
3
4  • insert into entrepots(nom_ent, adresse_ent, ville_ent, pays_ent)
5  values ("Bruxelles centre", "15 Rue Lamartine", "Bruxelles", "Belgique");
6
7  • insert into entrepots(nom_ent, adresse_ent, ville_ent, pays_ent)
8  values ("Milan Monza", "8 Avenue Garibaldi", "Milan", "Italie");
9
10 • insert into entrepots(nom_ent, adresse_ent, ville_ent, pays_ent)
11 values ("Paris 18ème", "26 Avenue Foch", "Paris", "France");
12
13 • insert into entrepots(nom_ent, adresse_ent, ville_ent, pays_ent)
14 values ("Berlin Ouest", "41 Rue Bismark", "Berlin", "Allemagne");
15
16 • select * from entrepots;
```

Result Grid					
Filter Rows:					
Edit:					
Export/Import:					
Wrap Cell Cont					
id ent	nom ent	adresse ent	ville ent	pays ent	
1	Lyon Est	2 Rue Jean Jaures	Lyon	France	
2	Bruxelles centre	15 Rue Lamartine	Bruxelles	Belgique	
3	Milan Monza	8 Avenue Garibaldi	Milan	Italie	
4	Paris 18ème	26 Avenue Foch	Paris	France	
5	Berlin Ouest	41 Rue Bismark	Berlin	Allemagne	

ntrepots 2 x



Même chose pour la table expéditions et voici le résultat :

	id_exp	date_exp	id_ent_source	id_ent_dest	poid	statut
▶	1	2023-11-03	2	4	325.50	Programme
	2	2023-11-01	1	5	540.00	En transit
	3	2023-10-30	4	2	152.75	En transit
	4	2023-10-25	3	1	430.20	Livrée
	5	2023-10-20	5	3	350.00	Livrée
	6	2023-11-03	3	5	275.65	Programme
	7	2023-11-01	2	1	125.00	En transit
	8	2023-10-22	4	3	489.30	Livrée
	9	2023-11-03	5	2	213.90	Programme
	10	2023-10-29	2	5	398.20	En transit
*	NULL	NULL	NULL	NULL	NULL	NULL

## Requêtes de base

Affichage de tous les entrepôts :

```
1 • select * from entrepots;
```

id_ent	nom_ent	adresse_ent	ville_ent	pays_ent
1	Lyon Est	2 Rue Jean Jaures	Lyon	France
2	Bruxelles centre	15 Rue Lamartine	Bruxelles	Belgique
3	Milan Monza	8 Avenue Garibaldi	Milan	Italie
4	Paris 18ème	26 Avenue Foch	Paris	France
5	Berlin Ouest	41 Rue Bissmark	Berlin	Allemagne
NULL	NULL	NULL	NULL	NULL

Affichage de toutes les expéditions :

```
1 • select * from expéditions;
```

id_exp	date_exp	id_ent_source	id_ent_dest	poid	statut
1	2023-11-03	2	4	325.50	Programme
2	2023-11-01	1	5	540.00	En transit
3	2023-10-30	4	2	152.75	En transit
4	2023-10-25	3	1	430.20	Livrée
5	2023-10-20	5	3	350.00	Livrée
6	2023-11-03	3	5	275.65	Programme
7	2023-11-01	2	1	125.00	En transit
8	2023-10-22	4	3	489.30	Livrée
9	2023-11-03	5	2	213.90	Programme
10	2023-10-29	2	5	398.20	En transit
NULL	NULL	NULL	NULL	NULL	NULL

Affichage de toutes les expéditions en transit :

```
1 • select * from expéditions
2 where statut = "En transit";
```

result Grid | Filter Rows:  | Edit: | Export/Import:

id_exp	date_exp	id_ent_source	id_ent_dest	poid	statut
2	2023-11-01	1	5	540.00	En transit
3	2023-10-30	4	2	152.75	En transit
7	2023-11-01	2	1	125.00	En transit
10	2023-10-29	2	5	398.20	En transit
NULL	NULL	NULL	NULL	NULL	NULL

Affichage de toutes les expéditions livrées :

```
1 • select * from expéditions
2 where statut = "Livrée";
```

Result Grid | Filter Rows:  | Edit: | Export/

id_exp	date_exp	id_ent_source	id_ent_dest	poid	statut
4	2023-10-25	3	1	430.20	Livrée
5	2023-10-20	5	3	350.00	Livrée
8	2023-10-22	4	3	489.30	Livrée
NULL	NULL	NULL	NULL	NULL	NULL

# Requêtes avancées

Afficher les entrepôts qui ont envoyé au moins une expédition en transit :

Le DISTINCT permet de ne sélectionner qu'une seule ligne à chaque fois et d'éviter les doublons. La jointure est externe car la clause WHERE est effectuée sur la table jointe et non sur la table dont on sélectionne les données, de plus elle se fait avec la clé étrangère id\_ent\_source pour n'avoir que les entrepôts qui ont **envoyés** une expédition en transit.

```
1 • SELECT DISTINCT e.nom_ent, e.adresse_ent, e.ville_ent, e.pays_ent
2 FROM entrepots AS e
3 LEFT JOIN expéditions AS ex ON e.id_ent = ex.id_ent_source
4 WHERE ex.statut = 'En transit';
```

nom_ent	adresse_ent	ville_ent	pays_ent
Lyon Est	2 Rue Jean Jaures	Lyon	France
Paris 18ème	26 Avenue Foch	Paris	France
Bruxelles centre	15 Rue Lamartine	Bruxelles	Belgique

Affichage des entrepôts qui ont reçu au moins une expédition en transit :

Le DISTINCT permet de ne sélectionner qu'une seule ligne à chaque fois et d'éviter les doublons. La jointure est externe car la clause WHERE est effectuée sur la table jointe et non sur la table dont on sélectionne les données, de plus elle se fait avec la clé étrangère id\_ent\_dest pour n'avoir que les entrepôts qui ont **reçu** une expédition en transit.

```
1 • SELECT DISTINCT e.nom_ent, e.adresse_ent, e.ville_ent, e.pays_ent
2 FROM entrepots AS e
3 LEFT JOIN expéditions AS ex ON e.id_ent = ex.id_ent_dest
4 WHERE ex.statut = 'En transit';
```

nom_ent	adresse_ent	ville_ent	pays_ent
Berlin Ouest	41 Rue Bismark	Berlin	Allemagne
Bruxelles centre	15 Rue Lamartine	Bruxelles	Belgique
Lyon Est	2 Rue Jean Jaures	Lyon	France

Affichage des expéditions qui ont un poids supérieur à 100 kg et qui sont en transit :

Double clause WHERE où l'on impose que le poids soit supérieur à 100 kilos et que le statut soit En transit.

```
1 • select *
2   from expéditions
3  where poids > 100
4  and statut = 'En transit';
```

Result Grid						
Filter Rows:						
Edit: Export/In						
id_exp	date_exp	id_ent_source	id_ent_dest	poids	statut	
2	2023-11-01	1	5	540.00	En transit	
3	2023-10-30	4	2	152.75	En transit	
7	2023-11-01	2	1	125.00	En transit	
10	2023-10-29	2	5	398.20	En transit	
NULL	NULL	NULL	NULL	NULL	NULL	

Affichage du nombre d'expéditions envoyées par chaque entrepôt :



Sélection du nom de chaque entrepôt et du COUNT de chaque id d'expédition que l'on transforme en alias nombre\_expéditions.

Jointure de la table expéditions pour avoir les id\_expéditions.

GROUP BY, pour chaque nom d'entrepôt unique dans la table entrepôts, le nombre total d'expéditions où cet entrepôt est l'entrepôt source (id\_ent\_source) sera compté et affiché.



```
1 • SELECT e.nom_ent, COUNT(ex.id_exp) AS nombre_expéditions
2   FROM entrepôts e
3  JOIN expéditions ex ON e.id_ent = ex.id_ent_source
4  GROUP BY e.nom_ent;
```

result Grid





Filter Rows:

Export:

Wrap Cell Content:



nom_ent	nombre_expeditions
Lyon Est	1
Bruxelles centre	3
Milan Monza	2
Paris 18ème	2
Berlin Ouest	2

Affichage du nombre total d'expéditions en transit :

COUNT qui prend le nombre de ligne dont le statut est en transit dans une colonne alias  
nombre\_exp\_transit

```
1 • select count(*) as nombre_exp_transit
2   from expéditions
3  where statut = 'En transit';
```

Result Grid		Filter Rows:	Export:
	nombre_exp_transit		
	4		

Affichage du nombre total d'expéditions livrées :

Même requête qu'au-dessus mais avec un statut livrée

```
1 • select count(*) as nombre_exp_livree
2   from expéditions
3  where statut = 'Livrée';
```

Result Grid		Filter Rows:	Export:	Wrap
	nombre_exp_livree			
	3			



Affichage du nombre total d'expéditions pour chaque mois de l'année en cours :



COUNT : compte le nombre total d'expéditions pour chaque groupe et le renomme en "NombreExpéditions"

WHERE : Cette condition filtre les enregistrements pour ne prendre que ceux de l'année en cours.

GROUP BY : Après avoir filtré les données pour l'année en cours, elles sont ensuite regroupées par année et mois

ORDER BY : Enfin, les résultats sont triés par mois

```
1 • SELECT YEAR(date_exp) AS Annee, MONTH(date_exp) AS Mois, COUNT(*) AS NombreExpéditions
2 FROM expéditions
3 WHERE YEAR(date_exp) = YEAR(CURDATE())
4 GROUP BY YEAR(date_exp), MONTH(date_exp)
5 ORDER BY Mois;
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	Annee	Mois	NombreExpéditions
▶	2023	10	5
	2023	11	5



Affichage des entrepôts qui ont envoyé des expéditions au cours des 30 derniers jours :

DISTINCT : Pour chaque entrepôt répondant aux critères, on ne le liste qu'une seule fois

INNER JOIN : On lie la table entrepots avec la table expéditions où l'entrepôt est le point de départ des expéditions

WHERE : On ne considère que les expéditions commencées dans les 30 jours précédant la date actuelle





```
1 • SELECT DISTINCT e.id_ent, e.nom_ent, e.adresse_ent, e.ville_ent, e.pays_ent
2 FROM entrepots e
3 INNER JOIN expéditions ex ON e.id_ent = ex.id_ent_source
4 WHERE ex.date_exp >= CURDATE() - INTERVAL 30 DAY;
```

Result Grid					
Filter Rows: <input type="text"/>					
Export:  Wrap Cell Content: 					
	id_ent	nom_ent	adresse_ent	ville_ent	pays_ent
•	2	Bruxelles centre	15 Rue Lamartine	Bruxelles	Belgique
	1	Lyon Est	2 Rue Jean Jaures	Lyon	France
	4	Paris 18ème	26 Avenue Foch	Paris	France
	3	Milan Monza	8 Avenue Garibaldi	Milan	Italie
	5	Berlin Ouest	41 Rue Bismark	Berlin	Allemagne

Affichage des entrepôts qui ont reçu des expéditions au cours des 30 derniers jours :

Même requête qu'au-dessus mais avec la réception et non l'envoi.

```
1 • SELECT DISTINCT e.id_ent, e.nom_ent, e.adresse_ent, e.ville_ent, e.pays_ent
2 FROM entrepots e
3 INNER JOIN expéditions ex ON e.id_ent = ex.id_ent_dest
4 WHERE ex.date_exp >= CURDATE() - INTERVAL 30 DAY;
```

Result Grid |   Filter Rows:  | Export:  | Wrap Cell Content: 

id_ent	nom_ent	adresse_ent	ville_ent	pays_ent
4	Paris 18ème	26 Avenue Foch	Paris	France
5	Berlin Ouest	41 Rue Bismark	Berlin	Allemagne
2	Bruxelles centre	15 Rue Lamartine	Bruxelles	Belgique
1	Lyon Est	2 Rue Jean Jaures	Lyon	France
3	Milan Monza	8 Avenue Garibaldi	Milan	Italie

# Requêtes complexes

Affichage des expéditions en transit qui ont été initiées par un entrepôt situé en Europe et à destination d'un entrepôt situé en Asie :

Je commence par ajouter une colonne continent\_ent qui va contenir le continent de chaque entrepôt

```
1 • alter table entrepots
2   add continent_ent varchar(50);
3
4 • update entrepots
5   set continent_ent = "Europe"
6   where pays_ent not like "Chine";
7
8 • update entrepots
9   set continent_ent = "Asie"
10  where pays_ent = "Chine";
```

id_ent	nom_ent	adresse_ent	ville_ent	pays_ent	continent_ent
1	Lyon Est	2 Rue Jean Jaures	Lyon	France	Europe
2	Beijing Aeroport	15 Rue Lamartine	Pékin	Chine	Asie
3	Milan Monza	8 Avenue Garibaldi	Milan	Italie	Europe
4	Paris 18ème	26 Avenue Foch	Paris	France	Europe
5	Berlin Ouest	41 Rue Bismark	Berlin	Allemagne	Europe
*	NULL	NULL	NULL	NULL	NULL

Grace à cette colonne, je peux mettre une clause WHERE qui impose que la source soit en Europe et la destination en Asie

```
1 • SELECT exp.*
2   FROM expéditions exp
3  JOIN entrepots ent_source ON exp.id_ent_source = ent_source.id_ent
4  JOIN entrepots ent_dest ON exp.id_ent_dest = ent_dest.id_ent
5  WHERE exp.statut = 'En transit'
6        AND ent_source.continent_ent = 'Europe'
7        AND ent_dest.continent_ent = 'Asie';
```

id_exp	date_exp	id_ent_source	id_ent_dest	poid	statut
3	2023-10-30	4	2	152.75	En transit

Affichage des entrepôts qui ont envoyé des expéditions à destination d'un entrepôt situé dans le même pays :

Simple SELECT avec une vérification que la source et la destination sont dans le même pays

```
1 • SELECT DISTINCT ent_source.nom_ent AS Entrepot_Expéditeur
2 FROM expéditions exp
3 JOIN entrepôts ent_source ON exp.id_ent_source = ent_source.id_ent
4 JOIN entrepôts ent_dest ON exp.id_ent_dest = ent_dest.id_ent
5 WHERE ent_source.pays_ent = ent_dest.pays_ent;
```

Result Grid

Entrepot_Expéditeur
Lyon Est

Affichage des entrepôts qui ont envoyé des expéditions à destination d'un entrepôt situé dans un pays différent :

Même requêtes qu'au-dessus mais on vérifie que la source et la destination ne soit pas dans le même pays

```
1 • SELECT DISTINCT ent_source.nom_ent AS Entrepot_Expéditeur
2 FROM expéditions exp
3 JOIN entrepôts ent_source ON exp.id_ent_source = ent_source.id_ent
4 JOIN entrepôts ent_dest ON exp.id_ent_dest = ent_dest.id_ent
5 WHERE ent_source.pays_ent != ent_dest.pays_ent;
```

Result Grid

Entrepot_Expéditeur
Beijing Aeroport
Milan Monza
Paris 18ème
Berlin Ouest

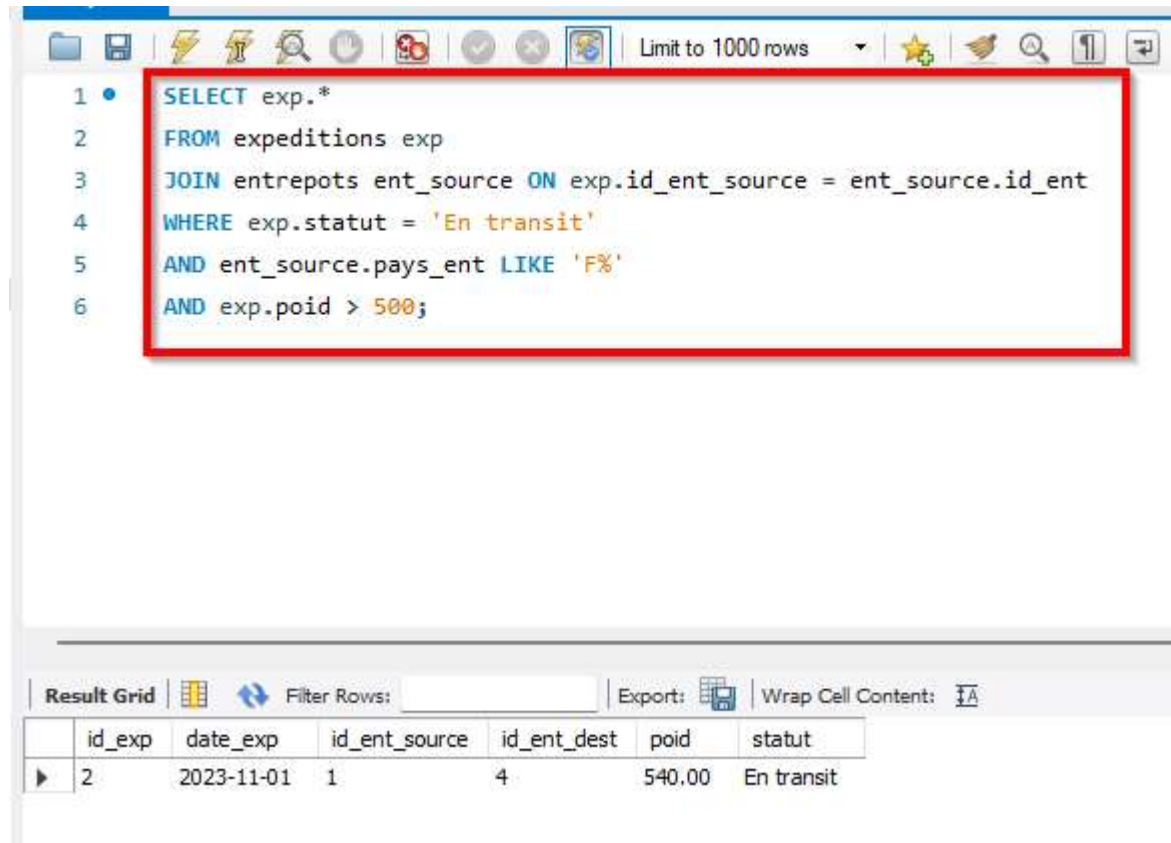


Affichage des expéditions en transit qui ont été initiées par un entrepôt situé dans un pays dont le nom commence par la lettre "F" et qui pèsent plus de 500 kg :

WHERE 1 : filtre est appliqué pour n'inclure que les expéditions dont le statut est "En transit"

WHERE 2 : filtre est appliqué pour n'inclure que les expéditions provenant d'entrepôts situés dans des pays dont le nom commence par "F" avec le %

WHERE 3 : filtre est appliqué pour n'inclure que les expéditions dont le poids est supérieur à 500



The screenshot shows a MySQL query editor window. The query is as follows:

```
1 • SELECT exp.*
2 FROM expéditions exp
3 JOIN entrepôts ent_source ON exp.id_ent_source = ent_source.id_ent
4 WHERE exp.statut = 'En transit'
5 AND ent_source.pays_ent LIKE 'F%'
6 AND exp.poid > 500;
```

The query is highlighted with a red box. Below the query editor, the 'Result Grid' is visible, showing the results of the query. The grid has columns: id\_exp, date\_exp, id\_ent\_source, id\_ent\_dest, poid, and statut. The first row of data is:

id_exp	date_exp	id_ent_source	id_ent_dest	poid	statut
2	2023-11-01	1	4	540.00	En transit

Affichage du nombre total d'expéditions pour chaque combinaison de pays d'origine et de destination :

```

1 • SELECT
2     ent_source.pays_ent AS Pays_Origine,
3     ent_dest.pays_ent AS Pays_Destination,
4     COUNT(*) AS Nombre_Expeditions
5 FROM
6     expéditions exp
7 INNER JOIN entrepôts ent_source ON exp.id_ent_source = ent_source.id_ent
8 INNER JOIN entrepôts ent_dest ON exp.id_ent_dest = ent_dest.id_ent
9 GROUP BY
10     Pays_Origine,
11     Pays_Destination;

```

Pays_Origine	Pays_Destination	Nombre_Expeditions
France	France	1
Chine	France	2
Chine	Allemagne	1
Italie	France	1
Italie	Allemagne	1
France	Chine	1
France	Italie	1
Allemagne	Italie	1
Allemagne	Chine	1

Affichage des entrepôts qui ont envoyé des expéditions au cours des 30 derniers jours et dont le poids total des expéditions est supérieur à 1000 kg :

```

1 • SELECT
2     ent_source.id_ent,
3     ent_source.nom_ent,
4     SUM(exp.poid) AS Poids_Total
5 FROM
6     expéditions exp
7 JOIN entrepôts ent_source ON exp.id_ent_source = ent_source.id_ent
8 WHERE
9     exp.date_exp >= CURDATE() - INTERVAL 30 DAY
10 GROUP BY
11     ent_source.id_ent,
12     ent_source.nom_ent
13 HAVING
14     SUM(exp.poid) > 1000;
15

```

id_ent	nom_ent	Poids_Total
1	Lyon Est	2000.00

Affichage des expéditions qui ont été livrées avec un retard de plus de 2 jours ouvrables :  
(pas réussis avec les jours ouvrables)

J'ai vu dans plusieurs forums qu'il était possible de créer un calendrier et de le déclarer dans ma requête et de supprimer les deux jours du week-end mais je n'ai pas réussi à le mettre en œuvre.

Cependant, je l'ai fait sans prendre en compte les jours non ouvrables :

Premièrement j'ajoute une colonne date\_livraison et une colonne date\_livraison\_prevu :

The screenshot shows a SQL IDE with a query editor and a result grid. The query editor contains the following SQL statements:

```
1 ALTER TABLE expéditions
2 ADD date_livraison DATE;
3
4 ALTER TABLE expéditions
5 ADD date_livraison_prevue DATE;
```

Red arrows point from the new columns in the SQL statements to the corresponding columns in the result grid. The result grid shows the following data:

	id_exp	date_exp	id_ent_source	id_ent_dest	poid	statut	date_livraison	date_livraison_prevue
1	2023-11-03	2	4	325,50	Programme	NULL	NULL	
2	2023-11-01	1	4	2000,00	En transit	NULL	NULL	
3	2023-10-30	4	2	152,75	En transit	NULL	NULL	
4	2023-10-25	3	1	430,20	Livrée	NULL	NULL	
5	2023-10-20	5	3	350,00	Livrée	NULL	NULL	
6	2023-11-03	3	5	275,65	Programme	NULL	NULL	
7	2023-11-01	2	1	125,00	En transit	NULL	NULL	
8	2023-10-22	4	3	489,30	Livrée	NULL	NULL	
9	2023-11-03	5	2	213,90	Programme	NULL	NULL	
10	2023-10-29	2	5	398,20	En transit	NULL	NULL	
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

Il n'y a plus qu'à calculer la différence entre les deux dates et en déduire le temps de retard

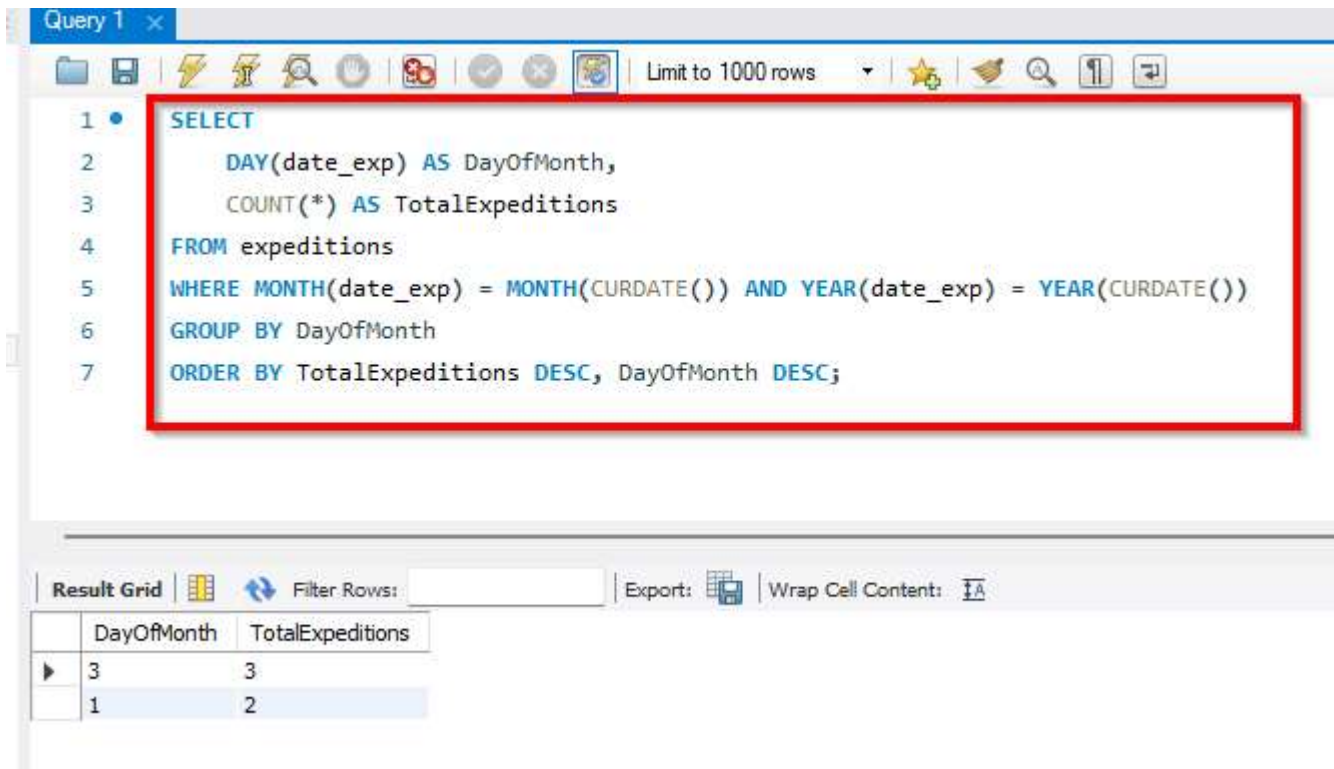
The screenshot shows a SQL IDE with a query editor and a result grid. The query editor contains the following SQL statement:

```
1 SELECT *
2 FROM expéditions
3 WHERE datediff(date_livraison, date_livraison_prevue) > 2;
```

The result grid shows the following data, with the row for id\_exp = 4 highlighted:

	id_exp	date_exp	id_ent_source	id_ent_dest	poid	statut	date_livraison	date_livraison_prevue
4	2023-10-25	3	1	430,20	Livrée	2023-11-02	2023-10-28	
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

Affichage du nombre total d'expéditions pour chaque jour du mois en cours, trié par ordre décroissant :



The screenshot shows a MySQL query editor window titled "Query 1". The query is as follows:

```
1 • SELECT
2     DAY(date_exp) AS DayOfMonth,
3     COUNT(*) AS TotalExpeditions
4 FROM expeditions
5 WHERE MONTH(date_exp) = MONTH(CURDATE()) AND YEAR(date_exp) = YEAR(CURDATE())
6 GROUP BY DayOfMonth
7 ORDER BY TotalExpeditions DESC, DayOfMonth DESC;
```

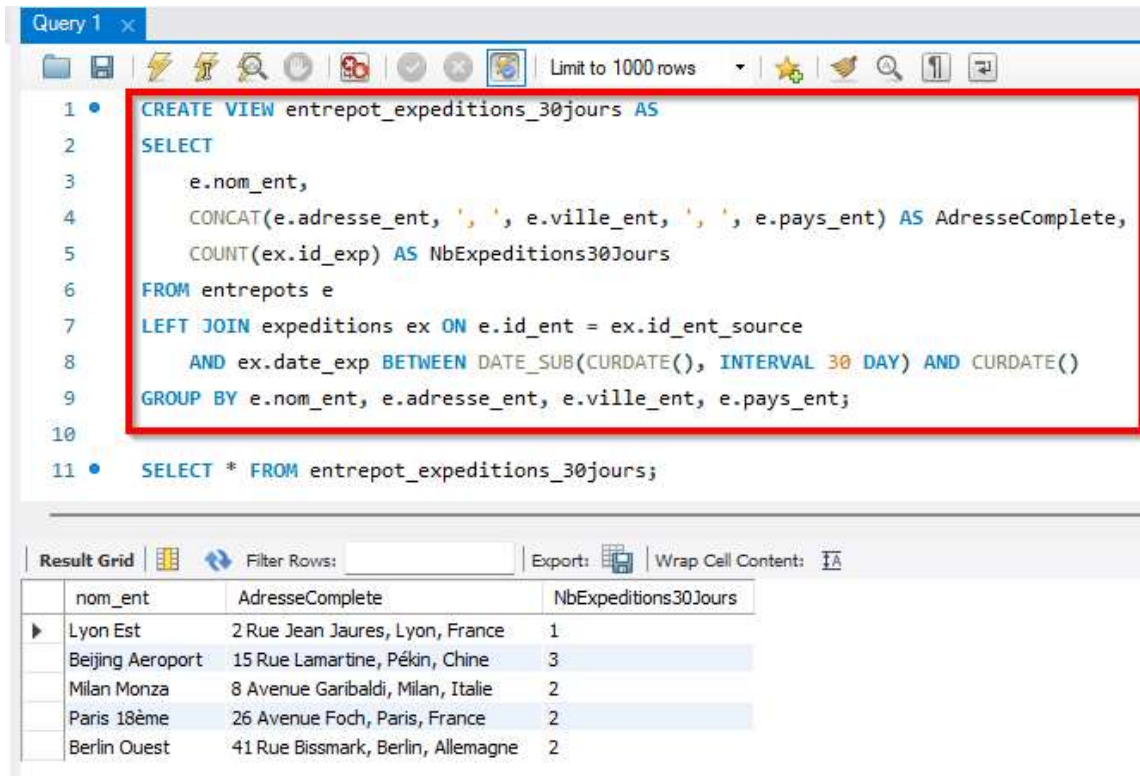
The query is highlighted with a red border. Below the query editor, the "Result Grid" is visible, showing the results of the query. The grid has two columns: "DayOfMonth" and "TotalExpeditions". The results are as follows:

DayOfMonth	TotalExpeditions
3	3
1	2



# T-SQL

Création d'une vue qui affiche les informations suivantes pour chaque entrepôt nom de l'entrepôt, adresse complète, nombre d'expéditions envoyées au cours des 30 derniers jours :



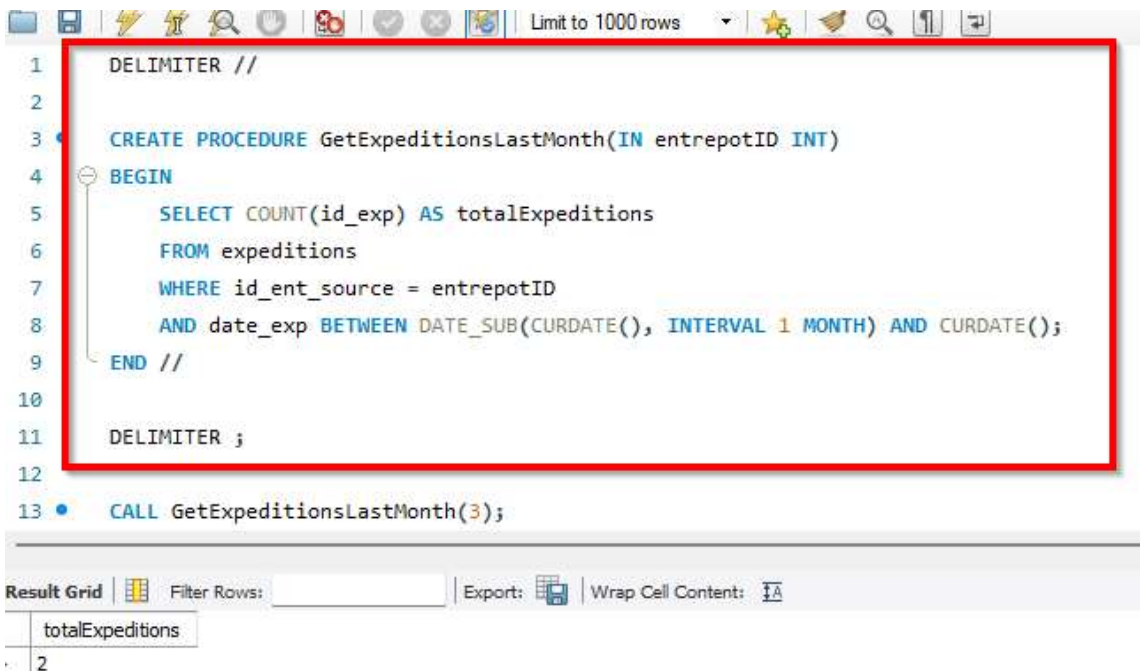
The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • CREATE VIEW entrepot_expeditions_30jours AS
2 SELECT
3     e.nom_ent,
4     CONCAT(e.adresse_ent, ', ', e.ville_ent, ', ', e.pays_ent) AS AdresseComplete,
5     COUNT(ex.id_exp) AS NbExpeditions30Jours
6 FROM entrepots e
7 LEFT JOIN expeditions ex ON e.id_ent = ex.id_ent_source
8     AND ex.date_exp BETWEEN DATE_SUB(CURDATE(), INTERVAL 30 DAY) AND CURDATE()
9 GROUP BY e.nom_ent, e.adresse_ent, e.ville_ent, e.pays_ent;
10
11 • SELECT * FROM entrepot_expeditions_30jours;
```

Below the query, the 'Result Grid' shows the following data:

nom_ent	AdresseComplete	NbExpeditions30Jours
Lyon Est	2 Rue Jean Jaures, Lyon, France	1
Beijing Aeroport	15 Rue Lamartine, Pékin, Chine	3
Milan Monza	8 Avenue Garibaldi, Milan, Italie	2
Paris 18ème	26 Avenue Foch, Paris, France	2
Berlin Ouest	41 Rue Bissmark, Berlin, Allemagne	2

Création d'une procédure stockée qui prend en entrée l'ID d'un entrepôt et renvoie le nombre total d'expéditions envoyées par cet entrepôt au cours du dernier mois :



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 DELIMITER //
2
3 • CREATE PROCEDURE GetExpeditionsLastMonth(IN entrepotID INT)
4 BEGIN
5     SELECT COUNT(id_exp) AS totalExpeditions
6     FROM expeditions
7     WHERE id_ent_source = entrepotID
8     AND date_exp BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 MONTH) AND CURDATE();
9 END //
10
11 DELIMITER ;
12
13 • CALL GetExpeditionsLastMonth(3);
```

Below the query, the 'Result Grid' shows the following data:

totalExpeditions
2

Création d'une fonction qui prend en entrée une date et renvoie le nombre total d'expéditions livrées ce jour-là :

```
1 DELIMITER //
```

```
2
```

```
3 CREATE FUNCTION GetTotalExpeditionsByDate(input_date DATE) RETURNS INT
```

```
4 DETERMINISTIC READS SQL DATA
```

```
5 BEGIN
```

```
6     DECLARE totalExpeditions INT;
```

```
7
```

```
8     SELECT COUNT(*)
```

```
9     INTO totalExpeditions
```

```
10    FROM expeditions
```

```
11    WHERE date_livraison = input_date AND statut = 'Livrée';
```

```
12
```

```
13    RETURN totalExpeditions;
```

```
14 END //
```

```
15
```

```
16 DELIMITER ;
```

```
17
```

```
18 • SELECT GetTotalExpeditionsByDate('2023-11-02');
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
GetTotalExpeditionsByDate('2023-11-02')				
1				