Labo 3 BDR

**-- Exercice 01**

SELECT

title,

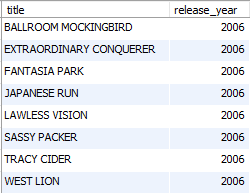
release\_year

FROM film

WHERE rating = "G" AND length > 100 AND replacement\_cost = 29.99

ORDER BY title;

-- END Exercice 01



**-- Exercice 02**

SELECT

customer\_id,

first\_name,

last\_name

FROM customer

WHERE first\_name = 'Tracy' AND store\_id = 1

ORDER BY customer\_id DESC;

-- END Exercice 02



**-- Exercice 03**

SELECT

customer\_id AS 'numéro',

first\_name AS 'prénom',

last\_name AS 'nom'

FROM customer

INNER JOIN address ON customer.address\_id = address.address\_id

WHERE active = 1 AND city\_id = 321 AND store\_id = 2

ORDER BY last\_name;

-- END Exercice 03



**-- Exercice 04**

SELECT

country AS 'pays',

city AS 'ville',

postal\_code AS 'npa'

FROM city

INNER JOIN country

ON city.country\_id = country.country\_id

JOIN address

ON city.city\_id = address.city\_id

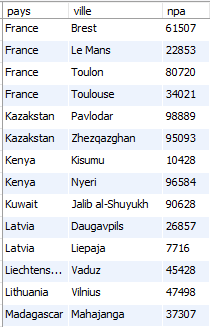
WHERE country = 'france'

OR country.country\_id > 50

AND country.country\_id < 58

ORDER BY country, city, npa;

-- END Exercice 04



**-- Exercice 05**

SELECT DISTINCT

film.film\_id,

film.title,

language.name AS 'langue'

FROM film

JOIN language

ON language.language\_id = film.language\_id

JOIN film\_actor

ON film\_actor.film\_id = film.film\_id

JOIN actor

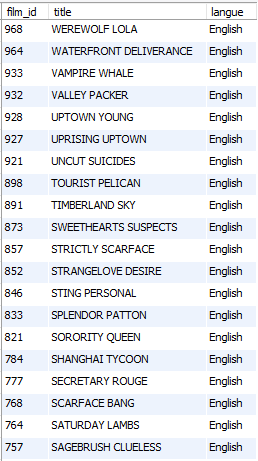
ON actor.actor\_id = film\_actor.actor\_id

WHERE actor.first\_name = 'alan'

OR actor.first\_name = 'ben'

ORDER BY film.film\_id DESC;

-- END Exercice 05



Remarque : Il y a 79 résultats en tout

**-- Exercice 06**

SELECT DISTINCT

c1.customer\_id AS 'customer1\_id',

c1.first\_name AS 'customer1\_first\_name',

c1.last\_name AS 'customer1\_last\_name',

c2.customer\_id AS 'customer2\_id',

c2.first\_name AS 'customer2\_first\_name',

c2.last\_name AS 'customer2\_last\_name'

FROM customer AS c1

INNER JOIN rental AS r1

ON c1.customer\_id = r1.customer\_id

INNER JOIN inventory AS i1

ON r1.inventory\_id = i1.inventory\_id

INNER JOIN inventory AS i2

ON i1.film\_id = i2.film\_id

INNER JOIN rental AS r2

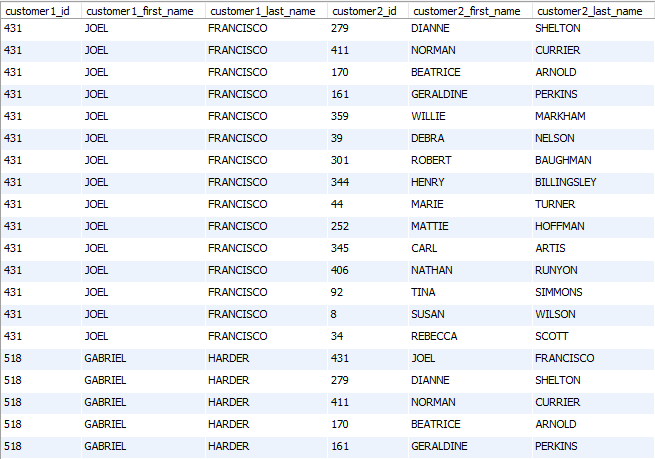
ON i2.inventory\_id = r2.inventory\_id

INNER JOIN customer AS c2

ON r2.customer\_id = c2.customer\_id

WHERE c1.customer\_id > c2.customer\_id;

-- END Exercice 06



Remarque : Il y a 98683 résultats en tout

**-- Exercice 07a**

SELECT DISTINCT

last\_name,

first\_name

FROM actor

INNER JOIN film\_actor

ON actor.actor\_id = film\_actor.actor\_id

INNER JOIN film\_category

ON film\_actor.film\_id = film\_category.film\_id

INNER JOIN category

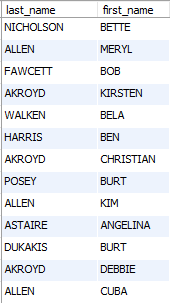
ON film\_category.category\_id = category.category\_id

WHERE category.name = 'Action'

AND first\_name LIKE 'B%'

OR last\_name LIKE 'A%';

-- END Exercice 07a



**-- Exercice 07b**

SELECT DISTINCT

last\_name,

first\_name

FROM actor, film\_actor, film\_category

WHERE actor.actor\_id = film\_actor.actor\_id

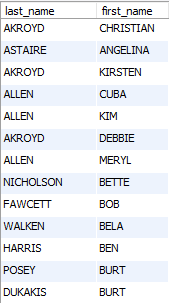
AND film\_actor.film\_id = film\_category.film\_id

AND film\_category.category\_id = 1

AND first\_name LIKE 'B%'

OR last\_name LIKE 'A%';

-- END Exercice 07b



**-- Exercice 08**

SELECT DISTINCT

c.customer\_id,

c.first\_name,

c.last\_name

FROM (

SELECT DISTINCT

customer.customer\_id,

customer.first\_name,

customer.last\_name,

film.title

FROM

customer

INNER JOIN rental

ON customer.customer\_id = rental.customer\_id

INNER JOIN inventory

ON rental.inventory\_id = inventory.inventory\_id

INNER JOIN film

ON inventory.film\_id = film.film\_id

WHERE film.film\_id

IN (SELECT DISTINCT film\_id

FROM actor

INNER JOIN film\_actor

ON actor.actor\_id = film\_actor.actor\_id

WHERE actor.first\_name = 'EMILY'

AND actor.last\_name = 'DEE'

)

) AS c

GROUP BY c.customer\_id

HAVING COUNT(c.customer\_id) = (

SELECT COUNT(\*)

FROM (

SELECT DISTINCT film\_id

FROM actor

INNER JOIN film\_actor

ON actor.actor\_id = film\_actor.actor\_id

WHERE actor.first\_name = 'EMILY'

AND actor.last\_name = 'DEE'

) AS film2

);

-- END Exercice 08



Remarque : Dans cet exercice, nous avons choisi de compter le nombre de films différents dans lesquels a joué l’actrice, et de comparer avec le nombre de films différents loués par les clients, dans lesquels a joué l’actrice.

**-- Exercice 09a**

SELECT

film.title AS 'titre',

COUNT(film\_actor.actor\_id) AS nombre\_acteurs

FROM film

INNER JOIN film\_actor

ON film.film\_id = film\_actor.film\_id

INNER JOIN film\_category

ON film.film\_id = film\_category.film\_id

INNER JOIN category

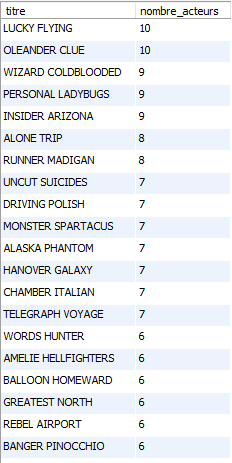
ON film\_category.category\_id = category.category\_id

WHERE category.name = 'Music'

GROUP BY title

ORDER BY nombre\_acteurs DESC;

-- END Exercice 09a



Remarque : Il y a 51 résultats en tout

**-- Exercice 09b**

SELECT

film.title AS 'titre',

COUNT(film\_actor.actor\_id) AS 'nombre\_acteurs'

FROM film

INNER JOIN film\_actor

ON film.film\_id = film\_actor.film\_id

INNER JOIN film\_category

ON film.film\_id = film\_category.film\_id

INNER JOIN category

ON film\_category.category\_id = category.category\_id

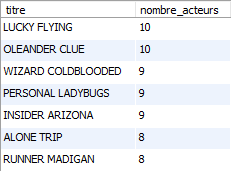
WHERE category.name = 'Music'

GROUP BY title

HAVING nombre\_acteurs > 7

ORDER BY nombre\_acteurs DESC;

-- END Exercice 09b



**-- Exercice 10**

SELECT

film\_category.category\_id AS 'id',

category.name AS 'nom',

COUNT(film\_id) AS 'nb\_films\_associés'

FROM film\_category

INNER JOIN category

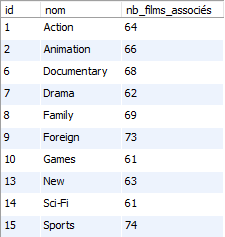
ON film\_category.category\_id = category.category\_id

GROUP BY film\_category.category\_id

HAVING nb\_films\_associés > 60

ORDER BY nom;

-- END Exercice 10



**-- Exercice 11**

SELECT

film.film\_id AS 'id\_min',

film.title AS 'titre\_min',

film.length AS 'durée\_min'

FROM film

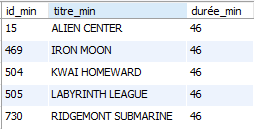
WHERE length = (

SELECT

MIN(length)

FROM film);

-- END Exercice 11



**-- Exercice 12**

SELECT

actor.actor\_id,

COUNT(film.film\_id) AS 'nombre\_films'

FROM actor

INNER JOIN film\_actor

ON actor.actor\_id = film\_actor.actor\_id

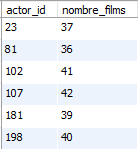
INNER JOIN film

ON film\_actor.film\_id = film.film\_id

GROUP BY actor\_id

HAVING nombre\_films > 35;

-- END Exercice 12



**-- Exercice 13a**

SELECT

film.film\_id AS 'id',

film.title AS 'titre'

FROM film

INNER JOIN film\_actor

ON film.film\_id = film\_actor.film\_id

WHERE film\_actor.actor\_id IN (

SELECT actor\_id

FROM (

SELECT

actor.actor\_id,

COUNT(film.film\_id) AS 'nombre\_films'

FROM actor

INNER JOIN film\_actor

ON actor.actor\_id = film\_actor.actor\_id

INNER JOIN film

ON film\_actor.film\_id = film.film\_id

GROUP BY actor\_id

HAVING nombre\_films > 35

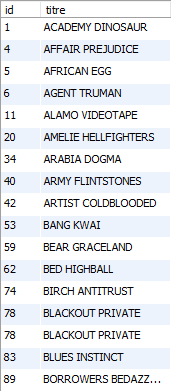
) AS a

)

AND film.film\_id < 100

ORDER BY id;

-- END Exercice 13a



**-- Exercice 13b**

SELECT

film.film\_id AS 'id',

film.title AS 'titre'

FROM film

INNER JOIN film\_actor

ON film.film\_id = film\_actor.film\_id

INNER JOIN (

SELECT

actor.actor\_id,

COUNT(film.film\_id) AS 'nombre\_films'

FROM actor

INNER JOIN film\_actor

ON actor.actor\_id = film\_actor.actor\_id

INNER JOIN film

ON film\_actor.film\_id = film.film\_id

GROUP BY actor\_id

HAVING nombre\_films > 35

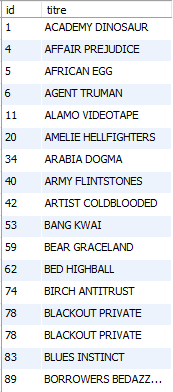
) AS a

ON film\_actor.actor\_id = a.actor\_id

WHERE film.film\_id < 100

ORDER BY id;

-- END Exercice 13b



Remarque : Les deux requêtes prennent exactement le même temps d’exécution sur nos ordis, ce qui est assez logique car il n’y a pas de différence fondamentale entre les deux : vérifier la présence d’un acteur dans une table (obtenue avec une sous-requête) ou faire une jointure avec cette même table revient à peu près au même. Faire une jointure peut être légèrement plus gourmand lorsqu’il y a beaucoup de colonnes, ce qui n’est pas le cas ici.

**-- Exercice 14**

SELECT

film.film\_id,

film.title,

film.rental\_rate AS 'prix'

FROM film

WHERE title NOT IN (

SELECT DISTINCT

film.title

FROM film

INNER JOIN inventory

ON film.film\_id = inventory.film\_id

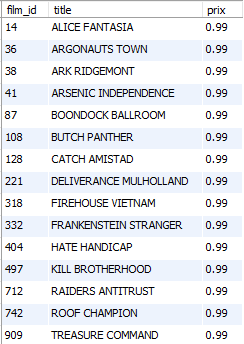
INNER JOIN rental

ON inventory.inventory\_id = rental.inventory\_id

)

AND film.rental\_rate < 2;

-- END Exercice 14



**-- Exercice 15**

SELECT

CEILING(SUM(length)/(60 \* 16)) AS jours

FROM film;

-- END Exercice 15



Remarque : Pour cet exercice, nous avons fait le choix d’arrondir à l’entier supérieur (ceiling), afin de ne pas avoir un nombre à virgule. La valeur exact en jours est : 120.075

**-- Exercice 16**

SELECT

customer.customer\_id AS 'id',

customer.last\_name AS 'nom',

customer.first\_name AS 'prénom',

country.country AS 'pays',

COUNT(customer.customer\_id) AS 'nombre\_films\_total',

SUM(payment.amount) AS 'total\_dépense',

AVG(payment.amount) AS 'dépense\_moyenne'

FROM customer

INNER JOIN payment

ON customer.customer\_id = payment.customer\_id

INNER JOIN address

ON customer.address\_id = address.address\_id

INNER JOIN city

ON address.city\_id = city.city\_id

INNER JOIN country

ON city.country\_id = country.country\_id

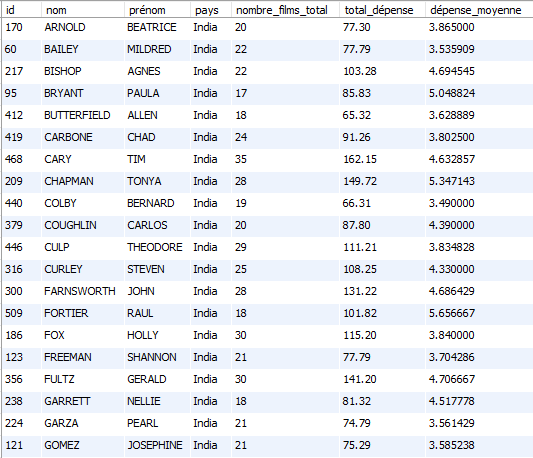
WHERE country.country IN ('India', 'Japan', 'Morocco')

GROUP BY customer.customer\_id

HAVING dépense\_moyenne > 3.4

ORDER BY pays, nom;

-- END Exercice 16



Remarque : Il y a 94 résultats en tout

**-- Exercice 17a**

SELECT DISTINCT

customer.customer\_id AS 'id',

customer.last\_name AS 'nom',

customer.first\_name AS 'prénom',

country.country AS 'pays'

FROM customer

INNER JOIN rental

ON customer.customer\_id = rental.customer\_id

INNER JOIN address

ON customer.address\_id = address.address\_id

INNER JOIN city

ON address.city\_id = city.city\_id

INNER JOIN country

ON city.country\_id = country.country\_id

WHERE EXISTS (

SELECT customer.customer\_id

FROM rental

WHERE customer.customer\_id = rental.customer\_id

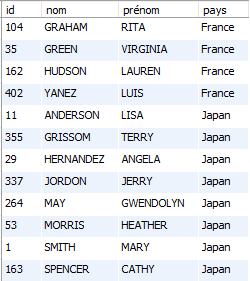
AND rental.return\_date IS NULL

)

AND country.country = 'Japan' OR country.country = 'France'

ORDER BY pays, nom;

-- END Exercice 17a



Remarque : Cet exercice ainsi que le 17b ne sont que des copies plus compliquées du 17c (qui est pour nous la façon la plus logique de procéder).

**-- Exercice 17b**

SELECT DISTINCT

customer.customer\_id AS 'id',

customer.last\_name AS 'nom',

customer.first\_name AS 'prénom',

country.country AS 'pays'

FROM customer

INNER JOIN rental

ON customer.customer\_id = rental.customer\_id

INNER JOIN address

ON customer.address\_id = address.address\_id

INNER JOIN city

ON address.city\_id = city.city\_id

INNER JOIN country

ON city.country\_id = country.country\_id

WHERE customer.customer\_id IN (

SELECT customer.customer\_id

FROM rental

WHERE customer.customer\_id = rental.customer\_id

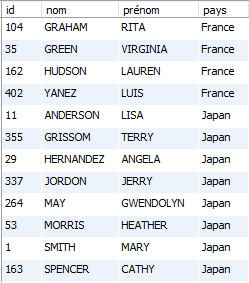
AND rental.return\_date IS NULL

)

AND country.country = 'Japan' OR country.country = 'France'

ORDER BY pays, nom;

-- END Exercice 17b



**-- Exercice 17c**

SELECT DISTINCT

customer.customer\_id AS 'id',

customer.last\_name AS 'nom',

customer.first\_name AS 'prénom',

country.country AS 'pays'

FROM customer

INNER JOIN rental

ON customer.customer\_id = rental.customer\_id

INNER JOIN address

ON customer.address\_id = address.address\_id

INNER JOIN city

ON address.city\_id = city.city\_id

INNER JOIN country

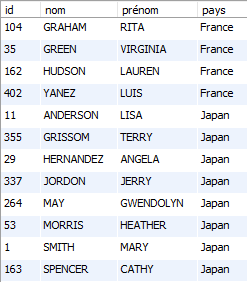
ON city.country\_id = country.country\_id

WHERE rental.return\_date IS NULL

AND country.country = 'Japan' OR country.country = 'France'

ORDER BY pays, nom;

-- END Exercice 17c



**-- Exercice 18a**

SELECT

COUNT(payment.amount)

FROM payment

WHERE payment.amount > 11;

-- END Exercice 18a



**-- Exercice 18b**

DELETE FROM payment

WHERE payment.amount > 11;

-- END Exercice 18b

**-- Exercice 18c**

SELECT

COUNT(payment.amount)

FROM payment

WHERE payment.amount > 11;

-- END Exercice 18c



**-- Exercice 19**

UPDATE payment

SET amount = amount \* 1.5, payment\_date = NOW()

WHERE amount > 5;

-- END Exercice 19

**-- Exercice 20**

DROP procedure if exists createMarcelRochat;

DELIMITER $$

create procedure createMarcelRochat()

begin

START transaction;

IF EXISTS (

SELECT country FROM country

WHERE country = "Switzerland")

THEN

update country set last\_update = now()

where country = "Switzerland";

ELSE

insert into country(country, last\_update)

values("Switzerland", now());

end if;

SELECT @country\_id := country\_id FROM country

WHERE country = "Switzerland";

IF EXISTS (

SELECT city FROM city

WHERE city = "Nyon")

THEN

update city set last\_update = now()

where city = "Nyon";

ELSE

insert into city(city, country\_id, last\_update)

values("Nyon", @country\_id, now());

end if;

SELECT

@city\_id :=city\_id

FROM city

WHERE city = "Nyon";

IF EXISTS (

SELECT address FROM address

WHERE address = "Rue du centre")

THEN

update address set last\_update = now()

where address = "Rue du centre";

ELSE

insert into address(address, district, city\_id, postal\_code, phone, last\_update)

values("Rue du centre", "Nyon", @city\_id, "1260", "022/360.00.00", now());

end if;

SELECT

@address\_id := address\_id FROM address

WHERE address = "Rue du centre";

IF EXISTS (

SELECT customer\_id FROM customer

WHERE first\_name = "Rochat" AND last\_name = "Marcel")

THEN

update customer set last\_update = now()

where first\_name = "Rochat" AND last\_name = "Marcel";

ELSE

insert into customer(store\_id, first\_name, last\_name, email, address\_id, active, create\_date, last\_update)

values(1, "Rochat", "Marcel", "mr@bluewin.ch", @address\_id, TRUE, now(), now());

end if;

COMMIT;

end$$

DELIMITER ;

CALL createMarcelRochat;

-- END Exercice 20

Question : La base génère l’id du client qui est une primary key, donc elle s’incrémente toute seule.

**-- Exercice 20d**

DROP procedure if exists testProcess;

DELIMITER $$

CREATE procedure testProcess()

begin

select \*

from customer

inner join address

on customer.address\_id = address. address\_id

inner join city

on address.city\_id = city.city\_id

inner join country

on city.country\_id = country.country\_id

where first\_name = "Rochat" AND last\_name = "Marcel";

end$$

DELIMITER ;

CALL testProcess;

-- END Exercice 20d



Remarque : Pour le programme de test, nous avons choisi de faire une jointure des 4 tables concernées (customer, address, city, country) et d’afficher le résultat correspondant au nom Rochat et prénom Marcel. Cette jointure des 4 tables contient toutes les colonnes et est donc très large.