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
```

title	release_year
BALLROOM MOCKINGBIRD	2006
EXTRAORDINARY CONQUERER	2006
FANTASIA PARK	2006
JAPANESE RUN	2006
LAWLESS VISION	2006
SASSY PACKER	2006
TRACY CIDER	2006
WEST LION	2006

-- 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
```

customer_id	first_name	last_name
589	TRACY	HERRMANN
108	TRACY	COLE

-- 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
```

numéro	prénom	nom
66	JANICE	WARD

-- 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
```

pays	ville	npa
France	Brest	61507
France	Le Mans	22853
France	Toulon	80720
France	Toulouse	34021
Kazakstan	Pavlodar	98889
Kazakstan	Zhezqazghan	95093
Kenya	Kisumu	10428
Kenya	Nyeri	96584
Kuwait	Jalib al-Shuyukh	90628
Latvia	Daugavpils	26857
Latvia	Liepaja	7716
Liechtens...	Vaduz	45428
Lithuania	Vilnius	47498
Madagascar	Mahajanga	37307

-- 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
```

film_id	title	langue
968	WEREWOLF LOLA	English
964	WATERFRONT DELIVERANCE	English
933	VAMPIRE WHALE	English
932	VALLEY PACKER	English
928	UPTOWN YOUNG	English
927	UPRISING UPTOWN	English
921	UNCUT SUICIDES	English
898	TOURIST PELICAN	English
891	TIMBERLAND SKY	English
873	SWEETHEARTS SUSPECTS	English
857	STRICTLY SCARFACE	English
852	STRANGELOVE DESIRE	English
846	STING PERSONAL	English
833	SPLENDOR PATTON	English
821	SORORITY QUEEN	English
784	SHANGHAI TYCOON	English
777	SECRETARY ROUGE	English
768	SCARFACE BANG	English
764	SATURDAY LAMBS	English
757	SAGEBRUSH CLUELESS	English

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
```

customer1_id	customer1_first_name	customer1_last_name	customer2_id	customer2_first_name	customer2_last_name
431	JOEL	FRANCISCO	279	DIANNE	SHELTON
431	JOEL	FRANCISCO	411	NORMAN	CURRIER
431	JOEL	FRANCISCO	170	BEATRICE	ARNOLD
431	JOEL	FRANCISCO	161	GERALDINE	PERKINS
431	JOEL	FRANCISCO	359	WILLIE	MARKHAM
431	JOEL	FRANCISCO	39	DEBRA	NELSON
431	JOEL	FRANCISCO	301	ROBERT	BAUGHMAN
431	JOEL	FRANCISCO	344	HENRY	BILLINGSLEY
431	JOEL	FRANCISCO	44	MARIE	TURNER
431	JOEL	FRANCISCO	252	MATTIE	HOFFMAN
431	JOEL	FRANCISCO	345	CARL	ARTIS
431	JOEL	FRANCISCO	406	NATHAN	RUNYON
431	JOEL	FRANCISCO	92	TINA	SIMMONS
431	JOEL	FRANCISCO	8	SUSAN	WILSON
431	JOEL	FRANCISCO	34	REBECCA	SCOTT
518	GABRIEL	HARDER	431	JOEL	FRANCISCO
518	GABRIEL	HARDER	279	DIANNE	SHELTON
518	GABRIEL	HARDER	411	NORMAN	CURRIER
518	GABRIEL	HARDER	170	BEATRICE	ARNOLD
518	GABRIEL	HARDER	161	GERALDINE	PERKINS

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
```

last_name	first_name
NICHOLSON	BETTE
ALLEN	MERYL
FAWCETT	BOB
AKROYD	KIRSTEN
WALKEN	BELA
HARRIS	BEN
AKROYD	CHRISTIAN
POSEY	BURT
ALLEN	KIM
ASTAIRE	ANGELINA
DUKAKIS	BURT
AKROYD	DEBBIE
ALLEN	CUBA

-- 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
```

last_name	first_name
AKROYD	CHRISTIAN
ASTAIRE	ANGELINA
AKROYD	KIRSTEN
ALLEN	CUBA
ALLEN	KIM
AKROYD	DEBBIE
ALLEN	MERYL
NICHOLSON	BETTE
FAWCETT	BOB
WALKEN	BELA
HARRIS	BEN
POSEY	BURT
DUKAKIS	BURT

-- 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
```

customer_id	first_name	last_name
1	MARY	SMITH

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
```

titre	nombre_acteurs
LUCKY FLYING	10
OLEANDER CLUE	10
WIZARD COLDBLOODED	9
PERSONAL LADYBUGS	9
INSIDER ARIZONA	9
ALONE TRIP	8
RUNNER MADIGAN	8
UNCUT SUICIDES	7
DRIVING POLISH	7
MONSTER SPARTACUS	7
ALASKA PHANTOM	7
HANOVER GALAXY	7
CHAMBER ITALIAN	7
TELEGRAPH VOYAGE	7
WORDS HUNTER	6
AMELIE HELLFIGHTERS	6
BALLOON HOMEWARD	6
GREATEST NORTH	6
REBEL AIRPORT	6
BANGER PINOCCHIO	6

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
```

titre	nombre_acteurs
LUCKY FLYING	10
OLEANDER CLUE	10
WIZARD COLDBLOODED	9
PERSONAL LADYBUGS	9
INSIDER ARIZONA	9
ALONE TRIP	8
RUNNER MADIGAN	8

-- 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
```

id	nom	nb_films_associés
1	Action	64
2	Animation	66
6	Documentary	68
7	Drama	62
8	Family	69
9	Foreign	73
10	Games	61
13	New	63
14	Sci-Fi	61
15	Sports	74

-- 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
```

id_min	titre_min	durée_min
15	ALIEN CENTER	46
469	IRON MOON	46
504	KWAI HOMEWARD	46
505	LABYRINTH LEAGUE	46
730	RIDGEMONT SUBMARINE	46

-- 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
```

actor_id	nombre_films
23	37
81	36
102	41
107	42
181	39
198	40

-- 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
```

id	titre
1	ACADEMY DINOSAUR
4	AFFAIR PREJUDICE
5	AFRICAN EGG
6	AGENT TRUMAN
11	ALAMO VIDEOTAPE
20	AMELIE HELLFIGHTERS
34	ARABIA DOGMA
40	ARMY FLINTSTONES
42	ARTIST COLDBLOODED
53	BANG KWAI
59	BEAR GRACELAND
62	BED HIGHBALL
74	BIRCH ANTITRUST
78	BLACKOUT PRIVATE
78	BLACKOUT PRIVATE
83	BLUES INSTINCT
89	BORROWERS BEDAZZ...

-- 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
```

id	titre
1	ACADEMY DINOSAUR
4	AFFAIR PREJUDICE
5	AFRICAN EGG
6	AGENT TRUMAN
11	ALAMO VIDEOTAPE
20	AMELIE HELLFIGHTERS
34	ARABIA DOGMA
40	ARMY FLINTSTONES
42	ARTIST COLDBLOODED
53	BANG KWAI
59	BEAR GRACELAND
62	BED HIGHBALL
74	BIRCH ANTITRUST
78	BLACKOUT PRIVATE
78	BLACKOUT PRIVATE
83	BLUES INSTINCT
89	BORROWERS BEDAZZ...

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
```

film_id	title	prix
14	ALICE FANTASIA	0.99
36	ARGONAUTS TOWN	0.99
38	ARK RIDGEMONT	0.99
41	ARSENIC INDEPENDENCE	0.99
87	BOONDOCK BALLROOM	0.99
108	BUTCH PANTHER	0.99
128	CATCH AMISTAD	0.99
221	DELIVERANCE MULHOLLAND	0.99
318	FIREHOUSE VIETNAM	0.99
332	FRANKENSTEIN STRANGER	0.99
404	HATE HANDICAP	0.99
497	KILL BROTHERHOOD	0.99
712	RAIDERS ANTITRUST	0.99
742	ROOF CHAMPION	0.99
909	TREASURE COMMAND	0.99

-- Exercice 15

```
SELECT
    CEILING(SUM(length)/(60 * 16)) AS jours
FROM film;
-- END Exercice 15
```

jours
121

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
```

id	nom	prénom	pays	nombre_films_total	total_dépense	dépense_moyenne
170	ARNOLD	BEATRICE	India	20	77.30	3.865000
60	BAILEY	MILDRED	India	22	77.79	3.535909
217	BISHOP	AGNES	India	22	103.28	4.694545
95	BRYANT	PAULA	India	17	85.83	5.048824
412	BUTTERFIELD	ALLEN	India	18	65.32	3.628889
419	CARBONE	CHAD	India	24	91.26	3.802500
468	CARY	TIM	India	35	162.15	4.632857
209	CHAPMAN	TONYA	India	28	149.72	5.347143
440	COLBY	BERNARD	India	19	66.31	3.490000
379	COUGHLIN	CARLOS	India	20	87.80	4.390000
446	CULP	THEODORE	India	29	111.21	3.834828
316	CURLEY	STEVEN	India	25	108.25	4.330000
300	FARNSWORTH	JOHN	India	28	131.22	4.686429
509	FORTIER	RAUL	India	18	101.82	5.656667
186	FOX	HOLLY	India	30	115.20	3.840000
123	FREEMAN	SHANNON	India	21	77.79	3.704286
356	FULTZ	GERALD	India	30	141.20	4.706667
238	GARRETT	NELLIE	India	18	81.32	4.517778
224	GARZA	PEARL	India	21	74.79	3.561429
121	GOMEZ	JOSEPHINE	India	21	75.29	3.585238

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
```

id	nom	prénom	pays
104	GRAHAM	RITA	France
35	GREEN	VIRGINIA	France
162	HUDSON	LAUREN	France
402	YANEZ	LUIS	France
11	ANDERSON	LISA	Japan
355	GRISSOM	TERRY	Japan
29	HERNANDEZ	ANGELA	Japan
337	JORDON	JERRY	Japan
264	MAY	GWENDOLYN	Japan
53	MORRIS	HEATHER	Japan
1	SMITH	MARY	Japan
163	SPENCER	CATHY	Japan

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
```

id	nom	prénom	pays
104	GRAHAM	RITA	France
35	GREEN	VIRGINIA	France
162	HUDSON	LAUREN	France
402	YANEZ	LUIS	France
11	ANDERSON	LISA	Japan
355	GRISSOM	TERRY	Japan
29	HERNANDEZ	ANGELA	Japan
337	JORDON	JERRY	Japan
264	MAY	GWENDOLYN	Japan
53	MORRIS	HEATHER	Japan
1	SMITH	MARY	Japan
163	SPENCER	CATHY	Japan

-- 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
```

id	nom	prénom	pays
104	GRAHAM	RITA	France
35	GREEN	VIRGINIA	France
162	HUDSON	LAUREN	France
402	YANEZ	LUIS	France
11	ANDERSON	LISA	Japan
355	GRISSOM	TERRY	Japan
29	HERNANDEZ	ANGELA	Japan
337	JORDON	JERRY	Japan
264	MAY	GWENDOLYN	Japan
53	MORRIS	HEATHER	Japan
1	SMITH	MARY	Japan
163	SPENCER	CATHY	Japan

-- Exercice 18a

```
SELECT
    COUNT(payment.amount)
```

```
FROM payment
WHERE payment.amount > 11;
-- END Exercice 18a
```

COUNT(payment.amount)
10

-- 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
```

COUNT(payment.amount)
0

-- 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
```

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
600	1	Rochat	Marcel	mr@bluewin.ch	606	1	2015-11-27 21:17:52	2015-11-27 2

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.