Base de Données - Labo 3

Simon Baehler et Sacha Bron - 17 novembre 2014

Exercice 1

Donner l'ensemble des films (`title`, `release_year`) classés (`rating`) G durant plus de 100 minutes, dont les coûts de remplacements sont 29.99$, en les ordonnant par titre.

Requête

```
SELECT
    `title`,
    `release_year`
FROM `film`
WHERE `rating` like 'G'
    AND `length` > 100
    AND `replacement_cost` = 29.99
ORDER BY `title`
```

Résultats (total: 8)

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 2

Donner le nom et le prénom (`first_name`, `last_name`) des clientes nommées TRACY attachées au magasin numéro 1, ordonnées par numéro de client décroissant

Requête

```
SELECT
    `customer_id`,
    `first_name`,
    `last_name`
FROM `customer`
WHERE `store_id` = 1
    AND `first_name` LIKE 'TRACY'
ORDER by `customer_id` DESC
```

Résultats (total: 2)

customer_id	first_name	last_name
589	TRACY	HERRMANN
108	TRACY	COLE

Exercice 3

Donner les films d'action (`film_id`, `title`) durant moins de 55 minutes en les ordonnant par `film_id` croissant.

Requête

```
SELECT
    F.film_id,
    `title`
FROM `film` AS F
    INNER JOIN `film_category` AS FC
        ON F.film_id = FC.film_id
    INNER JOIN category AS C
        ON FC.category_id = C.category_id
    WHERE c.name = 'action'
        AND f.length < 55
ORDER by F.film_id
```

Résultats (total: 6)

F.film_id	title
111	CADDYSHACK JEDI
292	EXCITEMENT EVE
542	LUST LOCK
697	PRIMARY GLASS
794	SIDE ARK
869	SUSPECTS QUILLS

Exercice 4

Donner les films (`film_id`, `title`, `langue`) de science fiction dans lesquels joue au moins un acteur dont le prénom est ALAN ou BEN en les ordonnant par numéro de film décroissant.

Requête

```
SELECT  DISTINCT
    F.film_id,
    `title`
FROM `film` AS F
    INNER JOIN `film_category` AS FC
```

```
        ON F.film_id = FC.film_id
    INNER JOIN category AS C
        ON FC.category_id = C.category_id
    INNER JOIN film_actor as AC
        ON F.film_id = AC.film_id
    INNER JOIN actor AS A
        ON AC.actor_id = A.actor_id
WHERE c.name = 'Sci-Fi'
    AND A.first_name = 'BEN'
    OR A.first_name = 'ALAN'
ORDER BY F.film_id DESC
```

Résultats (total: 31)

F.film_id	title
933	VAMPIRE WHALE
928	UPTOWN YOUNG
921	UNCUT SUICIDES
857	STRICTLY SCARFACE
852	STRANGELOVE DESIRE
757	SAGEBRUSH CLUELESS
753	RUSH GOODFELLAS
688	POLISH BROOKLYN
655	PANTHER REDS
592	MONSTER SPARTACUS
571	METAL ARMAGEDDON
564	MASSAGE IMAGE
550	MAGUIRE APACHE
529	LONELY ELEPHANT
496	KICK SAVANNAH
491	JUMPING WRATH
405	HAUNTED ANTITRUST
379	GREEDY ROOTS
369	GOODFELLAS SALUTE
313	FIDELITY DEVIL
235	DIVIDE MONSTER

Exercice 5

Donner les films (`film_id`, `title`, `langue`) de science fiction dans lesquels joue au moins un acteur dont leprénom est ALAN ou BEN en les ordonnant par numéro de film décroissant.

Requête

```
SELECT DISTINCT
    F.film_id ,
    `title`,
```

```
    `replacement_cost` AS `prix`
FROM `film` AS F
    LEFT JOIN inventory as I
        ON F.film_id = I.film_id
    LEFT JOIN rental as R
        ON I.inventory_id = R.inventory_id
WHERE `rental_id` IS NULL
    AND rental_rate < 2.00
```

Résultats (total: 16)

F.film_id	title	prix
1	ACADEMY DINOSAUR	20.99
14	ALICE FANTASIA	23.99
36	ARGONAUTS TOWN	12.99
38	ARK RIDGEMONT	25.99
41	ARSENIC INDEPENDENCE	17.99
87	BOONDOCK BALLROOM	14.99
108	BUTCH PANTHER	19.99
128	CATCH AMISTAD	10.99
221	DELIVERANCE MULHOLLAND	9.99
318	FIREHOUSE VIETNAM	14.99
332	FRANKENSTEIN STRANGER	16.99
404	HATE HANDICAP	26.99
497	KILL BROTHERHOOD	15.99
712	RAIDERS ANTITRUST	11.99
742	ROOF CHAMPION	25.99
909	TREASURE COMMAND	28.99

Exercice 6

Donner le nom et le prénom (`first_name`, `last_name`) des clients qui ont loué au moins une fois le même film.

Requête

```
SELECT DISTINCT
    C.first_name,
    C.last_name ,
    C1.first_name,
    C1.last_name
FROM `customer` AS C
    CROSS JOIN customer AS C1
    INNER JOIN rental as R
        ON R.customer_id = C.customer_id
    INNER JOIN inventory as I
        ON I.inventory_id = R.inventory_id
    INNER JOIN rental as R1
        ON R1.customer_id = C1.customer_id
```

```
    INNER JOIN inventory as I1
        ON I1.inventory_id = R1.inventory_id
WHERE R1.inventory_id = R.inventory_id
    AND C.customer_id < C1.customer_id
```

Résultats (total: 21510)

C.first_name	C.last_name	C1.first_name	C1.last_name
JOEL	FRANCISCO	GABRIEL	HARDER
DIANNE	SHELTON	JOEL	FRANCISCO
DIANNE	SHELTON	GABRIEL	HARDER
NORMAN	CURRIER	VIRGIL	WOFFORD
BEATRICE	ARNOLD	NORMAN	CURRIER
BEATRICE	ARNOLD	VIRGIL	WOFFORD
BEATRICE	ARNOLD	WILLIE	MARKHAM
GERALDINE	PERKINS	NORMAN	CURRIER
GERALDINE	PERKINS	BEATRICE	ARNOLD
GERALDINE	PERKINS	VIRGIL	WOFFORD
GERALDINE	PERKINS	WILLIE	MARKHAM
WILLIE	MARKHAM	NORMAN	CURRIER
WILLIE	MARKHAM	VIRGIL	WOFFORD
DEBRA	NELSON	DARREN	WINDHAM
ROBERT	BAUGHMAN	HENRY	BILLINGSLEY
SERGIO	STANFIELD	FREDDIE	DUGGAN
MARIE	TURNER	SERGIO	STANFIELD
MARIE	TURNER	FREDDIE	DUGGAN
MARIE	TURNER	MATTIE	HOFFMAN
MARIE	TURNER	DWAYNE	OLVERA

Exercice 7

Lister tous les clients actifs (`prenom`, `nom`) habitant la ville 321 et attachés au magasin 2. Trier par nom de famille.

Requête

```
SELECT
    `customer_id`,
    `first_name` AS `prenom`,
    `last_name` AS `nom`
FROM `customer` AS C
    INNER JOIN address as A
        ON C.address_id = A.address_id
    INNER JOIN store AS S
        ON C.store_id = S.store_id
WHERE A.city_id = 321
    AND S.store_id = 2
ORDER BY last_name
```

Résultats (total: 1)

customer_id	prenom	nom
66	JANICE	WARD

Exercice 8

Lister le pays, la ville, le numéro postal (`pays`, `ville`, `npa`) des villes françaises et des villes dont le numéro du pays auquel elles appartiennent est entre 50 et 58 (bornes non comprises). Ne pas utiliser `BETWEEN`. Ordonner par pays, ville, npa.

Requête

```
SELECT
    C.country AS `pays`,
    CI.city AS `ville`,
    A.postal_code AS `NPA`
FROM `city` AS CI
    INNER JOIN country AS C
        ON CI.country_id = C.country_id
    INNER JOIN address AS A
        ON A.city_id = CI.city_id
WHERE `country` LIKE 'France'
    OR CI.city_id >= 50 AND
    CI.city_id <= 58
ORDER BY `country`, CI.city, A.postal_code
```

Résultats (total: 13)

pays	ville	NPA
Brunei	Bandar Seri Begawan	52163
Cameroon	Bamenda	37636
France	Brest	61507
France	Le Mans	22853
France	Toulon	80720
France	Toulouse	34021
Gambia	Banjul	53446
India	Balurghat	89959
Israel	Bat Yam	62472
Switzerland	Basel	83980
Turkey	Balikesir	33050
Turkey	Batman	47753
Venezuela	Barcelona	15992

Exercice 9

Donner le nom et le prénom des acteurs ayant joué dans un film d'action, dont le prénom commence par b, ou dont le nom de famille commence par a. Donner deux versions de cette requêtes: en écrivant les jointures à l'aide du mot clé `JOIN`...

Requête

```
SELECT DISTINCT
    A.first_name,
    A.last_name
FROM `actor` AS A
    INNER JOIN film_actor AS FA
        ON A.actor_id = FA.actor_id
    INNER JOIN film_category AS FC
        ON FA.film_id = FC.film_id
    INNER JOIN category AS C
        ON FC.category_id = C.category_id
WHERE (`first_name` LIKE 'b%'
    OR `last_name` LIKE 'a%')
    AND c.name LIKE 'ACTION'
```

Résultats (total: 11)

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

Exercice 9b

...et sans le mot clé `JOIN`, en évitant les sous requêtes.

Requête

```
SELECT DISTINCT
    actor.first_name,
    actor.last_name
FROM
    actor,
    film_actor,
    film,
    film_category,
```

```
    category
WHERE
    actor.actor_id = film_actor.actor_id AND
    film_actor.film_id = film.film_id AND
     film.film_id = film_category.film_id AND
    film_category.category_id = category.category_id AND
    (actor.first_name LIKE 'b%'
    OR actor.last_name LIKE 'a%')
    AND category.name LIKE 'ACTION'
```

Résultats (total: 11)

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

Exercice 10

Donner le titre des films (`titre`) et le nombre d'acteurs (`nombre_acteurs`) des films de musique, en les triant par nombre d'acteur décroissant.

Requête

```
SELECT
    f.title AS titre,
    COUNT(fa.film_id) AS nombre_acteurs
FROM
    film f
JOIN film_actor fa ON fa.film_id = f.film_id
JOIN film_category fc ON fc.film_id = f.film_id
JOIN category c ON c.category_id = fc.category_id
WHERE c.name = 'Music'
GROUP BY f.film_id
ORDER BY nombre_acteurs DESC
```

Résultats (total: 997)

titre	nombre_acteurs
ACADEMY DINOSAUR	10
ACE GOLDFINGER	4

titre	nombre_acteurs
ADAPTATION HOLES	5
AFFAIR PREJUDICE	5
AFRICAN EGG	5
AGENT TRUMAN	7
AIRPLANE SIERRA	5
AIRPORT POLLOCK	4
ALABAMA DEVIL	9
ALADDIN CALENDAR	8
ALAMO VIDEOTAPE	4
ALASKA PHANTOM	7
ALI FOREVER	5
ALICE FANTASIA	4
ALIEN CENTER	6
ALLEY EVOLUTION	5
ALONE TRIP	8
ALTER VICTORY	4
AMADEUS HOLY	6
AMELIE HELLFIGHTERS	6

Exercice 11

Même question, mais la requête ne doit retourner que les films qui utilisent plus de 7 acteurs.

Requête

```
SELECT
    f.title AS titre,
    COUNT(fa.film_id) AS nombre_acteurs
FROM
    film f
JOIN film_actor fa ON fa.film_id = f.film_id
JOIN film_category fc ON fc.film_id = f.film_id
JOIN category c ON c.category_id = fc.category_id
WHERE c.name = 'Music'
GROUP BY f.film_id
HAVING nombre_acteurs >= 7
ORDER BY nombre_acteurs DESC
```

Résultats (total: 14)

titre	nombre_acteurs
LUCKY FLYING	10
OLEANDER CLUE	10
INSIDER ARIZONA	9
WIZARD COLDBLOODED	9
PERSONAL LADYBUGS	9

titre	nombre_acteurs
RUNNER MADIGAN	8
ALONE TRIP	8
HANOVER GALAXY	7
ALASKA PHANTOM	7
TELEGRAPH VOYAGE	7
UNCUT SUICIDES	7
CHAMBER ITALIAN	7
MONSTER SPARTACUS	7
DRIVING POLISH	7

Exercice 12

Lister les catégories (`id`, `nom`, `nombre de films associés`) de films associées à plus de 60 films, sans utiliser de sous requête. Ordonner les résultats par nom de catégorie.

Requête

```
SELECT
    c.category_id AS id,
    c.name AS nom,
     COUNT(fc.film_id) AS nombre_films
FROM
    category c
JOIN film_category fc ON fc.category_id = c.category_id
GROUP BY c.name
HAVING nombre_films >= 60
ORDER BY c.name
```

Résultats (total: 11)

id	nom	nombre_films
1	Action	64
2	Animation	66
3	Children	60
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 13

Afficher le film (ou les films si plusieurs films ont la même durée minimum) le plus court (`id_min`,

`titre_min`, `duree_min`).

Requête

```
SELECT
    f.film_id AS id_min,
    f.title AS titre_min,
    f.length AS duree_min
FROM film f
WHERE f.length = (
    SELECT
        MIN(f.length)
    FROM film f
)
```

Résultats (total: 5)

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

Exercice 14

Lister les acteurs (`actor_id`, `nombre_films`) qui ont joué dans plus de 35 films, sans utiliser de sous-requêtes.

Requête

```
SELECT
    a.actor_id AS actor_id,
    COUNT(fa.film_id) AS nombre_films
FROM actor a
JOIN film_actor fa ON fa.actor_id = a.actor_id
GROUP BY a.actor_id
HAVING nombre_films >= 35
```

Résultats (total: 12)

actor_id	nombre_films
13	35
23	37
37	35
60	35
81	36
102	41

actor_id	nombre_films
106	35
107	42
144	35
158	35
181	39
198	40

Exercice 15

Lister les films (`id`, `titre`) dont l'identifiant est inférieur à 100, ordonnés par id dans lesquels joue au moins un acteur qui a joué dans plus de 35 films.
Utiliser le mot clé `IN`.

Requête

```
SELECT
    f.film_id AS id,
    f.title AS titre
FROM film f
JOIN film_actor fa ON f.film_id = fa.film_id
JOIN actor a ON a.actor_id = fa.actor_id
WHERE f.film_id <= 100
    AND a.actor_id IN (
        SELECT
            a.actor_id AS actor_id
        FROM actor a
        JOIN film_actor fa ON fa.actor_id = a.actor_id
        GROUP BY a.actor_id
        HAVING COUNT(fa.film_id) >= 35
    )
ORDER BY id
```

Résultats (total: 37)

id	titre
1	ACADEMY DINOSAUR
4	AFFAIR PREJUDICE
5	AFRICAN EGG
6	AGENT TRUMAN
10	ALADDIN CALENDAR
11	ALAMO VIDEOTAPE
12	ALASKA PHANTOM
17	ALONE TRIP
18	ALTER VICTORY
19	AMADEUS HOLY
20	AMELIE HELLFIGHTERS
29	ANTITRUST TOMATOES

id	titre
31	APACHE DIVINE
32	APOCALYPSE FLAMINGOS
34	ARABIA DOGMA
40	ARMY FLINTSTONES
42	ARTIST COLDBLOODED
44	ATTACKS HATE
45	ATTRACTION NEWTON
47	BABY HALL

Exercice 16

Même question, mais sans utiliser le mot clé `IN`. Indication: une sous-requête peut être utilisée comme un table, et donc être jointe.

Requête

```
SELECT
    f.film_id AS id,
    f.title AS titre
FROM film f
JOIN film_actor fa ON f.film_id = fa.film_id
JOIN (
        SELECT
            a.actor_id AS actor_id
        FROM actor a
        JOIN film_actor fa ON fa.actor_id = a.actor_id
        GROUP BY a.actor_id
        HAVING COUNT(fa.film_id) >= 35
    ) a ON a.actor_id = fa.actor_id
WHERE f.film_id <= 100
ORDER BY f.film_id
```

Résultats (total: 37)

id	titre
1	ACADEMY DINOSAUR
4	AFFAIR PREJUDICE
5	AFRICAN EGG
6	AGENT TRUMAN
10	ALADDIN CALENDAR
11	ALAMO VIDEOTAPE
12	ALASKA PHANTOM
17	ALONE TRIP
18	ALTER VICTORY
19	AMADEUS HOLY
20	AMELIE HELLFIGHTERS

id	titre
29	ANTITRUST TOMATOES
31	APACHE DIVINE
32	APOCALYPSE FLAMINGOS
34	ARABIA DOGMA
40	ARMY FLINTSTONES
42	ARTIST COLDBLOODED
44	ATTACKS HATE
45	ATTRACTION NEWTON
47	BABY HALL

Question

Quelle requête est la plus rapide (celle-ci ou la précédente)? A votre avis, pourquoi?

Réponse

La requête la plus rapide sera celle-ci (la n°16). En effet, le DBMS n'aura pas besoin de charger 2 fois la table `actor` et de chercher si chaque actor est présent dans la table `actor` réduite. Toutes ces comparaisons permettent de gagner du temps.

Exercice 17

Un fou décide de regarder l'ensemble des films qui sont présents dans la base de données. Etablir une requête qui donne le nombre de jours (`jours`) qu'il devra y consacrer, s'il dispose de 16h par jour.

Requête

```
SELECT
    SUM(f.length)/60/16 AS jours
FROM film f
```

Résultats (total: 1)

```
120.07500000
```

Exercice 18

Afficher tous les clients résidant en Inde, au Japon, ou au Maroc, dont la dépense moyenne par film loué est supérieure à 3.4. Ordonner par pays puis par nom.
Afficher les informations suivantes: `id`, `nom`, `prenom`, `pays`, `nombre_films_total`, `total_depense`, `depense_moyenne`. Le coût de location est dans la table film.
Indication: Commencer par établir une requête affichant tous les clients avec leur dépense moyenne pour les films loués. Ensuite, créer une nouvelle requête qui ne retourne que les clients

dont la dépense moyenne par film est supérieure à 3.4, en utilisant la requête initiale comme sous-requête.

Requête

```
SELECT *
FROM (
    SELECT
        c.customer_id AS id,
        c.first_name AS prenom,
        c.last_name AS nom,
        co.country AS pays,
        COUNT(i.film_id) AS nombre_films_total,
        SUM(p.amount) AS total_depense,
        AVG(p.amount) AS depense_moyenne
    FROM customer c
    JOIN address ad ON ad.address_id = c.address_id
    JOIN city ON city.city_id = ad.city_id
    JOIN country co ON co.country_id = city.country_id
    JOIN payment p ON p.customer_id = c.customer_id
    JOIN rental r ON r.customer_id = c.customer_id
    JOIN inventory i ON i.inventory_id = r.inventory_id
    WHERE co.country = "India"
        OR co.country = "Morocco"
        OR co.country = "Japan"
    GROUP BY c.customer_id
    ) T
WHERE T.depense_moyenne > 3.4
ORDER BY T.pays, T.nom
```

Résultats (total: 93)

id	prenom	nom	pays	nombre_films_total	total_depense	depense_moyenne
170	BEATRICE	ARNOLD	India	676	3113.24	4.605385
60	MILDRED	BAILEY	India	625	2468.75	3.950000
217	AGNES	BISHOP	India	529	2271.71	4.294348
95	PAULA	BRYANT	India	324	1400.76	4.323333
412	ALLEN	BUTTERFIELD	India	441	1801.59	4.085238
419	CHAD	CARBONE	India	625	2243.75	3.590000
468	TIM	CARY	India	1521	6848.79	4.502821
209	TONYA	CHAPMAN	India	1024	5173.76	5.052500
440	BERNARD	COLBY	India	484	1953.16	4.035455
502	BRETT	CORNWELL	India	1156	4714.44	4.078235
379	CARLOS	COUGHLIN	India	529	2455.71	4.642174
446	THEODORE	CULP	India	961	3617.39	3.764194
316	STEVEN	CURLEY	India	841	3848.59	4.576207
300	JOHN	FARNSWORTH	India	961	4268.39	4.441613
509	RAUL	FORTIER	India	400	2016.00	5.040000
186	HOLLY	FOX	India	961	3555.39	3.699677

id	prenom	nom	pays	nombre_films_total	total_depense	depense_moyenne
123	SHANNON	FREEMAN	India	576	2418.24	4.198333
356	GERALD	FULTZ	India	900	3651.00	4.056667
238	NELLIE	GARRETT	India	441	1990.59	4.513810
224	PEARL	GARZA	India	484	1689.16	3.490000
121	JOSEPHINE	GOMEZ	India	676	2853.24	4.220769

Exercice 19

Donner la liste des clients japonais et français (`id`, `nom`, `prenom`, `pays`) qui n'ont pas encore rendu tous les films qu'ils ont empruntés.
Ordonner par pays, puis par nom. Utilisez `EXISTS`, ne pas utiliser de `GROUP BY`, ni de `IN` / `NOT IN`.

Requête

```
SELECT
    `customer_id` AS `id`,
    `first_name` AS `prenom`,
    `last_name` AS `nom`,
    CO.country  AS `pays`
FROM `customer` AS C
    INNER JOIN address AS A
        ON C.address_id = A.address_id
    INNER JOIN city AS CI
        ON A.city_id = CI.city_id
    INNER JOIN country AS CO
        ON CI.country_id = CO.country_id
WHERE EXISTS
    (SELECT *
    FROM rental AS R
    WHERE R.customer_id = C.customer_id
        AND R.return_date IS NULL)
    AND (Co.country LIKE 'Japan' OR CO.country LIKE 'France')
ORDER BY pays, nom
```

Résultats (total: 8)

id	prenom	nom	pays
162	LAUREN	HUDSON	France
11	LISA	ANDERSON	Japan
355	TERRY	GRISSOM	Japan
29	ANGELA	HERNANDEZ	Japan
337	JERRY	JORDON	Japan
264	GWENDOLYN	MAY	Japan
53	HEATHER	MORRIS	Japan
163	CATHY	SPENCER	Japan

Exercice 20

Même question. Utiliser `IN`, ne pas utiliser de `GROUP BY`, ni de `EXISTS` / `NOT EXISTS`.

Requête

```
SELECT
    customer_id AS id,
    first_name AS prenom,
    last_name AS nom,
    CO.country AS pays
FROM customer AS C
    INNER JOIN address AS A
        ON C.address_id = A.address_id
    INNER JOIN city AS CI
        ON A.city_id = CI.city_id
    INNER JOIN country AS CO
        ON CI.country_id = CO.country_id
WHERE C.customer_id IN
    (SELECT C.customer_id
        FROM rental AS R
        WHERE R.customer_id = C.customer_id
            AND R.return_date IS NULL)
    AND (Co.country LIKE 'Japan' OR CO.country LIKE 'France')
ORDER BY pays, nom
```

Résultats (total: 8)

id	prenom	nom	pays
162	LAUREN	HUDSON	France
11	LISA	ANDERSON	Japan
355	TERRY	GRISSOM	Japan
29	ANGELA	HERNANDEZ	Japan
337	JERRY	JORDON	Japan
264	GWENDOLYN	MAY	Japan
53	HEATHER	MORRIS	Japan
163	CATHY	SPENCER	Japan

Exercice 21

Même question. Ne pas utiliser de `GROUP BY`, de `IN` / `NOT IN`, ni de `EXISTS` / `NOT EXISTS`.

Requête

```
SELECT DISTINCT
    c.customer_id AS id,
    first_name AS prenom,
    last_name AS nom,
    co.country  AS pays
FROM customer AS c
```

```
    INNER JOIN address AS a
        ON c.address_id = a.address_id
    INNER JOIN city
        ON a.city_id = city.city_id
    INNER JOIN country AS co
        ON city.country_id = co.country_id
     LEFT JOIN rental AS R
        ON r.customer_id = c.customer_id
          WHERE r.return_date IS NULL AND (co.country LIKE 'Japan' OR co.country
LIKE 'France')
ORDER BY pays, nom
```

Résultats (total: 8)

id	prenom	nom	pays
162	LAUREN	HUDSON	France
11	LISA	ANDERSON	Japan
355	TERRY	GRISSOM	Japan
29	ANGELA	HERNANDEZ	Japan
337	JERRY	JORDON	Japan
264	GWENDOLYN	MAY	Japan
53	HEATHER	MORRIS	Japan
163	CATHY	SPENCER	Japan

Exercice 22

Lister le nombre de paiements dont la valeur supérieure est à 11. Effacer ces paiements. Lister à nouveau pour vérifier que l'opération a bien eu lieu. Donner les trois requêtes et les résultats de la première et de la troisième.

Requêtes

```
SELECT COUNT(p.payment_id)
FROM payment p
WHERE p.amount > 11;
DELETE FROM payment
WHERE amount > 11;
SELECT COUNT(p.payment_id)
FROM payment p
WHERE p.amount > 11;
```

Exercice 23

En une seule requête, modifier les paiements comme suit: Chaque paiement de plus de 5 est majoré de 50% et la date de paiement est mise à jour à la date courante du serveur.

Requête

```
UPDATE payment
SET amount = (amount + amount * 0.5), payment_date = CURRENT_TIMESTAMP
WHERE amount > 5
```

Exercice 24

Insérez le nouveau client actif dans la base, avec toutes les informations requises pour que vous puissiez louer des films. Spécifier les attributs (colonnes) lors de l'insertion.
Indications: plusieurs requêtes sont nécessaires. Pour chaque nouveau tuple, la base de données doit générer l'id.

Requête

```
INSERT INTO
    city (city, country_id)
VALUES ('Nyon', (
        SELECT
            country_id AS C
    FROM country
    WHERE country='Switzerland'));
INSERT INTO
    address (address,city_id,postal_code,phone,district)
VALUES ("Rue du centre",(
        SELECT city_id FROM city
        WHERE city LIKE "Nyon"),
            1260,"022 360 00 00","");
INSERT INTO
    customer (store_id,first_name,last_name,email,address_id,active,create_date)
VALUES (1,"Marcel","Rochat","mr@bluewin.ch",(
        SELECT
            max(address_id)
        FROM address),1,CURRENT_DATE());
SELECT C.first_name,
        C.last_name,
        A.address,
        A.postal_code,
        CI.city,
        A.phone,
        CO.country,
        C.email,
        C.store_id
FROM customer AS C
    INNER JOIN address AS A
        ON C.address_id = A.address_id
    INNER JOIN city AS CI
        ON A.city_id = CI.city_id
    INNER JOIN country AS CO
        ON CI.country_id = CO.country_id
WHERE first_name = "MARCEL" AND last_name = "ROCHAT";
```

Résultats (total: 1)

C.first_name	C.last_name	A.address	A.postal_code	CI.city	A.phone	CO.country	C.email	C.store_id
Marcel	Rochat	Rue du centre	1260	Nyon	022 360 00 00	Switzerland	mr@bluewin.ch	1

Question

Pourquoi ne pouvez-vous pas le faire? Nyon n'existe pas, il faut donc la crée.

Réponse

Nous pourrions générer l'id des tuples ajoutés, mais il faudrait au préalable faire une requête pour trouver l'id le plus grand de la table.
Cette opération est fastidieuse et peut provoquer des conflits si deux clients essayent d'insérer des tuples en même temps.

Exercice 25

Requête