




## Exercice 1

Donnez le numéro, le nom et l'email ( `customer_id` , `nom` , `email` ) des clients dont le prénom est **PHYLLIS**, qui sont rattachés au magasin numéro 1, ordonnés par numéro de client décroissant.

### Requête

```
SELECT
    customer_id,
    last_name,
    email
FROM customer
WHERE store_id=1
      AND first_name='PHYLLIS'
ORDER BY customer_id DESC;
```

### Résultats (total: 1)

	 customer_id ▾	 last_name ▾	 email ▾
1	93	FOSTER	PHYLLIS.FOSTER@sakilacustomer.org

## Exercice 2

Donnez l'ensemble des films ( `titre` , `annee_sortie` ) classés ( `rating` ) R, ayant une durée de moins de 60 minutes et dont les coûts de remplacements sont 12.99\$, en les ordonnant par titre.

### Requête

```
SELECT title, release_year
FROM film AS f
WHERE f.rating = 'R'
      AND f.length < 60
      AND f.replacement_cost = 12.99
ORDER BY f.title;
```

### Résultats (total: 2)

	title	release_year
1	CLOSER BANG	2006
2	GO PURPLE	2006

## Exercice 3

Listez le pays, la ville et le numéro postal ( `country` , `city` , `postal_code` ) des villes française, ainsi que des villes dont le numéro de pays est entre 63 et 67 (bornes comprises), en les ordonnant par pays puis par ville et finalement par code postal. **N'utilisez pas de BETWEEN.**

## Requête

```
SELECT
    c.country,
    ci.city,
    a.postal_code
FROM country AS c
JOIN city AS ci
    ON ci.country_id = c.country_id
JOIN address AS a
    ON a.city_id = ci.city_id
WHERE c.country='France'
    OR (c.country_id >= 63 AND c.country_id <= 67)
ORDER BY c.country, ci.city, a.postal_code;
```

## Résultats (total: 16)

	country	city	postal_code
1	France	Brest	61507
2	France	Le Mans	22853
3	France	Toulon	80720
4	France	Toulouse	34021
5	Mozambique	Beira	86768
6	Mozambique	Naala-Porto	65892
7	Mozambique	Tete	71986
8	Myanmar	Monywa	68146
9	Myanmar	Myingyan	53561
10	Nauru	Yangor	65952
11	Nepal	Birgunj	97960
12	Netherlands	Amersfoort	33711
13	Netherlands	Apeldoorn	11044
14	Netherlands	Ede	504
15	Netherlands	Emmen	42123
16	Netherlands	s-Hertogenbosch	43310

## Exercice 4

Listez tous les clients actifs ( `customer_id`, `prenom`, `nom` ) habitant la ville 171, et rattachés au magasin numéro 1. Triez-les par ordre alphabétique des prénoms

## Requête

```
SELECT customer_id, last_name, first_name
FROM customer
JOIN address a
    ON customer.address_id = a.address_id
WHERE store_id = 1
    AND a.city_id = 171
ORDER BY first_name;
```

**Résultats (total: 1)**

	customer_id	last_name	first_name
1	51	STEWART	ALICE

## Exercice 5

Donnez le nom et le prénom ( `prenom_1`, `nom_1`, `prenom_2`, `nom_2` ) des clients qui ont loué au moins une fois le même film (par exemple, si **ALAN** et **BEN** ont loué le film **MATRIX**, mais pas **TRACY**, seuls **ALAN** et **BEN** doivent être listés).

## Requête

```
SELECT DISTINCT
  c1.first_name AS prenom_1,
  c1.last_name AS nom_1,
  c2.first_name AS prenom_2,
  c2.last_name AS nom_2
FROM rental AS r1
JOIN customer AS c1
  ON r1.customer_id = c1.customer_id
JOIN inventory AS i1
  ON r1.inventory_id = i1.inventory_id
JOIN inventory AS i2
  ON i1.film_id = i2.film_id
JOIN rental AS r2
  ON i2.inventory_id = r2.inventory_id
JOIN customer AS c2
  ON r2.customer_id = c2.customer_id
  AND r1.customer_id < r2.customer_id
WHERE (c1.first_name, c1.last_name) != (c2.first_name, c2.last_name);
```

**Résultats (total: 98'692)**

	🔍 prenom_1 ⚙	🔍 nom_1 ⚙	🔍 prenom_2 ⚙	🔍 nom_2 ⚙
1	JAVIER	ELROD	HUGH	WALDROP
2	TAMARA	NGUYEN	BRADLEY	MOTLEY
3	BONNIE	HUGHES	ANGEL	BARCLAY
4	KIM	CRUZ	ALEX	GRESHAM
5	ANGELA	HERNANDEZ	MILDRED	BAILEY
6	CONSTANCE	REID	BILLY	POULIN
7	GREGORY	MAULDIN	BARRY	LOVELACE
8	JOANNE	ROBERTSON	RICHARD	MCCRARY
9	TIMOTHY	BUNN	KARL	SEAL
10	FRANCES	PARKER	CARMEN	OWENS
11	DANNY	ISOM	SHANE	MILLARD
12	LISA	ANDERSON	ANNETTE	OLSON
13	SHELLY	WATTS	JUSTIN	NGO
14	GLEN	TALBERT	EDUARDO	HIATT
15	LAUREN	HUDSON	JUSTIN	NGO
16	MELINDA	FERNANDEZ	SONIA	GREGORY
17	WENDY	HARRISON	TED	BREAUX
18	BRANDON	HUEY	STEVE	MACKENZIE
19	LINDA	WILLIAMS	ROBIN	HAYES
20	RUTH	MARTINEZ	KURT	EMMONS

## Exercice 6

Donnez le nom et le prénom des acteurs ( `nom` , `prenom` ) ayant joué dans un film d'horreur, dont le prénom commence par K, ou dont le nom de famille commence par D **sans utiliser le mot clé JOIN**.

## Requête

```
SELECT first_name, last_name
FROM actor,
      film,
      film_actor,
      film_category,
      category
WHERE actor.actor_id = film_actor.actor_id
      AND film_actor.film_id = film.film_id
```

```
AND film_category.film_id = film.film_id  
AND film_category.category_id = category.category_id  
AND category.name LIKE 'Horror'  
AND (actor.last_name LIKE 'D%' OR actor.first_name LIKE 'K%');
```

**Résultats (total: 49)**

	🔍 first_name	🔍 last_name
1	CHRIS	DEPP
2	JODIE	DEGENERES
3	SCARLETT	DAMON
4	KENNETH	PESCI
5	SUSAN	DAVIS
6	LUCILLE	DEE
7	KENNETH	TORN
8	KARL	BERRY
9	JUDY	DEAN
10	KARL	BERRY
11	KENNETH	TORN
12	BURT	DUKAKIS
13	LUCILLE	DEE
14	KENNETH	TORN
15	SUSAN	DAVIS
16	CHRIS	DEPP
17	GINA	DEGENERES
18	KENNETH	HOFFMAN
19	ALAN	DREYFUSS
20	KENNETH	TORN

## Exercice 7

Donnez les films ( `id`, `titre`, `prix_de_location_par_jour` ) dont le prix de location par jour est inférieur ou égal à 1.00\$ et qui n'ont jamais été loués. Écrire la requête de 2 différentes façons

(changer les clauses pour exprimer l'exclusion).

*Indication : Le prix de location par jour doit être calculé.*

## Requête 1

```
SELECT
    f.film_id,
    f.title,
    (f.rental_rate / f.rental_duration) AS prix_location_par_jour
FROM film AS f
WHERE (f.rental_rate / f.rental_duration) <= 1.00
    AND f.film_id NOT IN(
        SELECT
            film_id
        FROM inventory
        WHERE inventory_id IN(
            SELECT inventory_id
            FROM rental
        )
    );
```

## Résultats 1 (total: 36)



	film_id ↕	title ↕	prix_location_par_jour ↕
1	14	ALICE FANTASIA	0.165
2	33	APOLLO TEEN	0.598
3	36	ARGONAUTS TOWN	0.14142857142857142857
4	38	ARK RIDGEMONT	0.165
5	41	ARSENIC INDEPENDENCE	0.2475
6	87	BOONDOCK BALLROOM	0.14142857142857142857
7	108	BUTCH PANTHER	0.165
8	128	CATCH AMISTAD	0.14142857142857142857
9	144	CHINATOWN GLADIATOR	0.71285714285714285714
10	148	CHOCOLATE DUCK	0.99666666666666666667
11	171	COMMANDMENTS EXPRESS	0.83166666666666666667
12	198	CRYSTAL BREAKING	0.49833333333333333333
13	217	DAZED PUNK	0.83166666666666666667
14	221	DELIVERANCE MULHOLLAND	0.2475
15	318	FIREHOUSE VIETNAM	0.14142857142857142857
16	325	FLOATS GARDEN	0.49833333333333333333
17	332	FRANKENSTEIN STRANGER	0.14142857142857142857
18	359	GLADIATOR WESTWARD	0.83166666666666666667
19	404	HATE HANDICAP	0.2475
20	419	HOCUS FRIDA	0.7475

## Requête 2

```

SELECT
    f.film_id,
    f.title,
    (f.rental_rate / f.rental_duration) AS prix_location_par_jour
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 (f.rental_rate / f.rental_duration) <= 1.00 AND i.inventory_id IS NULL;

```

## Résultats 2 (total: 36)

	film_id ↕	title ↕	prix_location_par_jour ↕
1	802	SKY MIRACLE	0.42714285714285714286
2	497	KILL BROTHERHOOD	0.2475
3	801	SISTER FREDDY	0.998
4	359	GLADIATOR WESTWARD	0.83166666666666666667
5	325	FLOATS GARDEN	0.49833333333333333333
6	33	APOLLO TEEN	0.598
7	198	CRYSTAL BREAKING	0.49833333333333333333
8	419	HOCUS FRIDA	0.7475
9	332	FRANKENSTEIN STRANGER	0.14142857142857142857
10	712	RAIDERS ANTITRUST	0.2475
11	404	HATE HANDICAP	0.2475
12	642	ORDER BETRAYED	0.42714285714285714286
13	954	WAKE JAWS	0.71285714285714285714
14	221	DELIVERANCE MULHOLLAND	0.2475
15	108	BUTCH PANTHER	0.165
16	701	PSYCHO SHRUNK	0.598
17	128	CATCH AMISTAD	0.14142857142857142857
18	38	ARK RIDGEMONT	0.165
19	318	FIREHOUSE VIETNAM	0.14142857142857142857
20	669	PEARL DESTINY	0.99666666666666666667

## Exercice 8

Donnez la liste des clients (id, nom, prenom) espagnols qui n'ont pas encore rendu tous les films qu'ils ont empruntés, en les ordonnant par nom. Donnez 3 versions de cette requête :

1. En utilisant **EXISTS** (pas de **GROUP BY**, ni de **IN** ou **NOT IN**).
2. En utilisant **IN** (pas de **GROUP BY**, ni de **EXISTS** ou **NOT EXISTS**).
3. En utilisant aucun des mot-clés précédent (c'est à dire pas de **GROUP BY**, **IN**, **NOT IN**, **EXISTS**, **NOT EXISTS**).

## Requête 1

```
SELECT customer.customer_id, customer.first_name, customer.last_name
FROM customer
      JOIN address a
        ON customer.address_id = a.address_id
      JOIN city c1
        ON a.city_id = c1.city_id
      JOIN country c2
        ON c1.country_id = c2.country_id
```

```
WHERE c2.country LIKE 'Spain'
AND EXISTS(SELECT *
            FROM rental
            WHERE rental.customer_id = customer.customer_id
               AND rental.return_date IS NULL);
```

## Résultats 1 (total: 2)

	customer_id	first_name	last_name
1	52	JULIE	SANCHEZ
2	394	CHRIS	BROTHERS

## Requête 2

```
SELECT
    customer.customer_id,
    customer.first_name,
    customer.last_name
FROM customer
JOIN address a
    ON customer.address_id = a.address_id
JOIN city c1
    ON a.city_id = c1.city_id
JOIN country c2
    ON c1.country_id = c2.country_id
WHERE c2.country LIKE 'Spain'
AND customer.customer_id IN (
    SELECT rental.customer_id
    FROM rental
    WHERE return_date IS NULL
);
```

## Résultats 2 (total: 2)

	customer_id	first_name	last_name
1	52	JULIE	SANCHEZ
2	394	CHRIS	BROTHERS

## Requête 3

```

SELECT customer.customer_id, customer.first_name, customer.last_name
FROM customer
    JOIN address a
        ON customer.address_id = a.address_id
    JOIN city c1
        ON a.city_id = c1.city_id
    JOIN country c2
        ON c1.country_id = c2.country_id
    JOIN rental r
        ON customer.customer_id = r.customer_id
WHERE c2.country LIKE 'Spain'
    AND r.return_date IS NULL;

```

## Résultats 3 (total: 2)

	customer_id	first_name	last_name
1	52	JULIE	SANCHEZ
2	394	CHRIS	BROTHERS

## Exercice 9 - BONUS

Donnez le numéro, le nom et le prénom ( `customer_id`, `prenom`, `nom` ) des clients qui ont loué tous les films de l'actrice **EMILY DEE**.

## Requête

```

SELECT
    c.customer_id,
    c.first_name,
    c.last_name
FROM customer AS c
JOIN rental AS r
    ON c.customer_id = r.customer_id
JOIN inventory AS i
    ON r.inventory_id = i.inventory_id
JOIN film AS f
    ON i.film_id = f.film_id
JOIN film_actor AS fa
    ON f.film_id = fa.film_id
JOIN actor AS a
    ON fa.actor_id = a.actor_id
WHERE a.first_name = 'EMILY'

```

```

    AND a.last_name = 'DEE'
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING COUNT(DISTINCT f.film_id) = (
    SELECT COUNT(*)
    FROM film AS films
    JOIN film_actor AS fa
      ON films.film_id = fa.film_id
    JOIN actor AS actors
      ON fa.actor_id = actors.actor_id
   WHERE actors.first_name = 'EMILY'
      AND actors.last_name = 'DEE'
);

```

## Résultats (total: 1)

	customer_id	first_name	last_name
1	562	WALLACE	SLONE

## Exercice 10

Donnez le titre des films et le nombre d'acteurs ( `titre`, `nb_acteurs` ) des films dramatiques en les triant par le nombre d'acteur décroissant. Retenez uniquement les films avec moins de 5 acteurs.

## Requête

```

SELECT film.title, count(fa.actor_id) as nb_acteurs
FROM film
    JOIN film_category fc
      ON film.film_id = fc.film_id
    JOIN film_actor fa
      ON film.film_id = fa.film_id
    JOIN category c
      ON fc.category_id = c.category_id
WHERE c.name LIKE 'Drama'
GROUP BY film.title
HAVING count(fa.actor_id) >= 5
ORDER BY nb_acteurs DESC

```

## Résultats (total: 41)

	📄 title ↕	📄 nb_acteurs ↕
1	CHITTY LOCK	13
2	SPICE SORORITY	11
3	SAINTS BRIDE	10
4	JACKET FRISCO	10
5	CONEHEADS SMOOCHY	9
6	WEST LION	9
7	SCORPION APOLLO	8
8	WARDROBE PHANTOM	8
9	UNFAITHFUL KILL	8
10	GREEDY ROOTS	8
11	SEA VIRGIN	8
12	NECKLACE OUTBREAK	8
13	HOBBIT ALIEN	8
14	BUNCH MINDS	8
15	VIRGIN DAISY	8
16	APOLLO TEEN	8
17	HAROLD FRENCH	7
18	REUNION WITCHES	7
19	HANGING DEEP	7
20	LIES TREATMENT	7

## Exercice 11

Listez les catégories ( `id` , `nom` , `nb_films` ) de films associées à plus de 65 films, sans utiliser de sous-requête, et en les ordonnant par nombre de films.

## Requête

```
SELECT
    c.category_id,
    c.name,
    COUNT(fc.film_id) AS nb_film
FROM category AS c
JOIN film_category AS fc
    ON c.category_id = fc.category_id
GROUP BY c.category_id, c.name
HAVING COUNT(fc.film_id) > 65
ORDER BY nb_film;
```

## Résultats (total: 5)

	category_id ↕	name ↕	nb_film ↕
1	2	Animation	66
2	6	Documentary	68
3	8	Family	69
4	9	Foreign	73
5	15	Sports	74

## Exercice 12

Affichez le(s) film(s) (id, titre, duree) ayant la durée la moins longue. Si plusieurs films ont la même durée (la moins longue), il faut afficher l'ensemble de ces derniers.

## Requête

```
SELECT
    film_id AS id,
    title AS titre,
    length AS duree
FROM
    film
WHERE
    length = (
        SELECT MIN(length)
```

```
FROM film
);
```

## Résultats (total: 5)

	id	titre	duree
1	15	ALIEN CENTER	46
2	469	IRON MOON	46
3	504	KWAI HOMEWARD	46
4	505	LABYRINTH LEAGUE	46
5	730	RIDGEMONT SUBMARINE	46

## Exercice 13

Listez les film ( id , titre ) dans lesquels jouent au moins un acteur qui a joué dans plus de 40 films, en les ordonnant par titre. Donnez 2 versions de cette requête :

1. En utilisant le mot-clé **IN**
2. Sans utiliser le mot-clé **IN**

*Indication : une sous-requête peut être utilisée comme une table, sur laquelle on peut donc effectuer des jointures.*



## Requête 1

```
SELECT DISTINCT
f.film_id,
  f.title
FROM film AS f
JOIN film_actor AS fa
  ON f.film_id = fa.film_id
WHERE fa.actor_id IN (
  SELECT
    a.actor_id
  FROM actor AS a
  LEFT JOIN film_actor AS fa
    ON a.actor_id = fa.actor_id
  GROUP BY a.actor_id
  HAVING COUNT(fa.film_id) > 40
)
```



```
HAVING COUNT(fa.actor_id) > 40
);
```

## Résultats 1 (total: 81)

	 film_id	 title
1	162	CLUELESS BUCKET
2	34	ARABIA DOGMA
3	172	CONEHEADS SMOOCHY
4	431	HOOSIERS BIRDCAGE
5	600	MOTIONS DETAILS
6	879	TELEGRAPH VOYAGE
7	277	ELEPHANT TROJAN
8	165	COLDBLOODED DARLING
9	849	STORM HAPPINESS
10	138	CHARIOTS CONSPIRACY
11	781	SEVEN SWARM
12	112	CALENDAR GUNFIGHT
13	409	HEARTBREAKERS BRIGHT
14	326	FLYING HOOK
15	239	DOGMA FAMILY
16	133	CHAMBER ITALIAN
17	837	STAGE WORLD
18	548	MAGNIFICENT CHITTY
19	392	HALL CASSIDY
20	603	MOVIE SHAKESPEARE

## Requête 2

```
SELECT DISTINCT
    f.film_id,
    f.title
FROM film f
JOIN film_actor fa
    ON f.film_id = fa.film_id
JOIN (
    SELECT
        a.actor_id
    FROM actor a
    JOIN film_actor fa
        ON a.actor_id = fa.actor_id
    GROUP BY a.actor_id
    HAVING COUNT(fa.actor_id) > 40
) sub
    ON fa.actor_id = sub.actor_id
ORDER BY f.title;
```

## Résultats 2 (total: 81)

	film_id	title
1	20	AMELIE HELLFIGHTERS
2	34	ARABIA DOGMA
3	53	BANG KWAI
4	62	BED HIGHBALL
5	112	CALENDAR GUNFIGHT
6	123	CASABLANCA SUPER
7	124	CASPER DRAGONFLY
8	133	CHAMBER ITALIAN
9	136	CHAPLIN LICENSE
10	138	CHARIOTS CONSPIRACY
11	162	CLUELESS BUCKET
12	165	COLDBLOODED DARLING
13	172	CONEHEADS SMOOCHY
14	194	CROW GREASE
15	200	CURTAIN VIDEOTAPE
16	205	DANCES NONE
17	209	DARKNESS WAR
18	220	DEER VIRGINIAN
19	239	DOGMA FAMILY
20	268	EARLY HOME

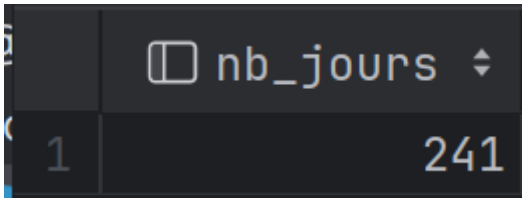
## Exercice 14

Un fou furieux décide de regarder l'ensemble des films qui sont présents dans la base de données. Etablissez une requête qui donne le nombre de jours (nb\_jours) qu'il devra y consacrer sachant qu'il dispose de 8 h par jour.

## Requête

```
SELECT
    CEIL(SUM(length) / 60.0 / 8.0) AS nb_jours
FROM
    film;
```

## Résultats (total: 1)



	nb_jours
1	241

## Exercice 15

Affichez tous les clients ( id , nom , email , pays , nb\_locations , depense\_totale , depense\_moyenne ) résidant en Suisse, en France ou en Allemagne, dont la dépense moyenne (montant payé) par location est supérieure à 3.0, en les ordonnant par pays puis par nom.

*Indication : Commencez par établir une requête affichant tous les clients avec leur dépense moyenne pour les films loués. Ensuite, une nouvelle requête qui ne retourne que les clients dont la dépense moyenne par film est supérieure à 3.0, en utilisant la première requête comme sous-requête.*

## Requête

```
SELECT
    id_customer,
    nom_customer,
    email_customer,
    country_customer,
    nb_locations,
    depense_totale,
    depense_moyenne
FROM (
    SELECT
        c.customer_id AS id_customer,
        c.first_name AS nom_customer,
        c.email AS email_customer,
        co.country AS country_customer ,
        COUNT(r.customer_id) AS nb_locations,
        SUM(p.amount) AS depense_totale,
```

```

        AVG(p.amount) AS depense_moyenne
FROM customer c
JOIN rental r
        ON c.customer_id = r.customer_id
JOIN address a
        ON c.address_id = a.address_id
JOIN city ci
        ON a.city_id = ci.city_id
JOIN country co
        ON ci.country_id = co.country_id
JOIN payment p
        ON c.customer_id = p.customer_id
GROUP BY c.customer_id, c.first_name, c.email, co.country
)
WHERE country_customer IN ('Switzerland', 'France', 'Germany')
AND depense_moyenne > 3.0
ORDER BY country_customer, nom_customer;

```

## Résultats (total: 14)

	id_customer	nom_customer	email_customer	country...	nb_locations	depense_totale	depense_moyenne
1	162	LAUREN	LAUREN.HUDSON@sakilacustomer.org	France	400	1436	3.59
2	402	LUIS	LUIS.YANEZ@sakilacustomer.org	France	400	1596	3.99
3	104	RITA	RITA.GRAHAM@sakilacustomer.org	France	576	2226.24	3.865
4	35	VIRGINIA	VIRGINIA.GREEN@sakilacustomer.org	France	1024	4149.76	4.0525
5	196	ALMA	ALMA.AUSTIN@sakilacustomer.org	Germany	1225	5307.75	4.3328571428571429
6	227	COLLEEN	COLLEEN.BURTON@sakilacustomer.org	Germany	576	2106.24	3.6566666666666667
7	114	GRACE	GRACE.ELLIS@sakilacustomer.org	Germany	1089	4609.11	4.2324242424242424
8	448	MIGUEL	MIGUEL.BETANCOURT@sakilacustomer.org	Germany	841	3935.59	4.6796551724137931
9	195	VANESSA	VANESSA.SIMS@sakilacustomer.org	Germany	361	1649.39	4.5689473684210526
10	201	VICKI	VICKI.FIELDS@sakilacustomer.org	Germany	625	2718.75	4.35
11	251	VICKIE	VICKIE.BREWER@sakilacustomer.org	Germany	961	3741.39	3.8932258064516129
12	155	GAIL	GAIL.KNIGHT@sakilacustomer.org	Switzerland	625	2743.75	4.39
13	61	KATHERINE	KATHERINE.RIVERA@sakilacustomer.org	Switzerland	196	824.04	4.2042857142857143
14	598	WADE	WADE.DELVALLE@sakilacustomer.org	Switzerland	484	1843.16	3.8081818181818182

## Exercice 16

Donnez les 3 requêtes suivantes ainsi que le résultat de la première et de la dernière :

1. Comptez les paiements dont la valeur est inférieure ou égale à 9.
2. Effacez ces mêmes paiements.
3. Comptez à nouveau ces mêmes paiements pour vérifier que l'opération a bien eu lieu.

## Requête 1

```

SELECT count(*)
FROM payment
WHERE amount <= 9;

```

## Résultats 1 (total: 1)

	count ↕
1	15678

## Requête 2

```
DELETE FROM payment
WHERE amount <= 9
RETURNING *;
```

## Résultats 2(total: 15'678)

	payment_id ↕	customer_id ↕	staff_id ↕	rental_id ↕	amount ↕	payment_date ↕
1	16050	269	2	7	1.99	2017-01-24 21:40:19.996577 +00:00
2	16051	269	1	98	0.99	2017-01-25 15:16:50.996577 +00:00
3	16052	269	2	678	6.99	2017-01-28 21:44:14.996577 +00:00
4	16053	269	2	703	0.99	2017-01-29 00:58:02.996577 +00:00
5	16054	269	1	750	4.99	2017-01-29 08:10:06.996577 +00:00
6	16055	269	2	1099	2.99	2017-01-31 12:23:14.996577 +00:00
7	16056	270	1	193	1.99	2017-01-26 05:10:14.996577 +00:00
8	16057	270	1	1040	4.99	2017-01-31 04:03:42.996577 +00:00
9	16058	271	1	1096	8.99	2017-01-31 11:59:15.996577 +00:00
10	16059	272	1	33	0.99	2017-01-25 02:47:17.996577 +00:00
11	16060	272	1	405	6.99	2017-01-27 12:01:05.996577 +00:00
12	16061	272	1	1041	6.99	2017-01-31 04:14:49.996577 +00:00
13	16062	272	1	1072	0.99	2017-01-31 08:21:16.996577 +00:00
14	16063	273	2	122	3.99	2017-01-25 18:14:47.996577 +00:00
15	16064	273	2	980	0.99	2017-01-30 20:13:45.996577 +00:00
16	16065	274	1	147	2.99	2017-01-25 22:46:16.996577 +00:00
17	16066	274	1	208	4.99	2017-01-26 06:38:48.996577 +00:00
18	16067	274	2	301	2.99	2017-01-26 19:34:40.996577 +00:00
19	16068	274	1	394	5.99	2017-01-27 09:54:37.996577 +00:00
20	16069	274	2	474	2.99	2017-01-27 20:40:22.996577 +00:00

## Requête 3

```
SELECT count(*)
FROM payment
WHERE amount <= 9;
```

## Résultats 3 (total: 1)

	count
1	0

## Exercice 17

En une seule requête, modifiez les paiements comme suit :

1. Chaque paiement de plus de 4\$ est majoré de 50 %.
2. La date de paiement est mise à jour avec la date courante du serveur.

## Requête

```
UPDATE payment
SET amount = amount * 1.5,
    payment_date = now()
WHERE amount > 4
RETURNING *;
```

## Résultats (total: 371)

	payment_id	customer_id	staff_id	rental_id	amount	payment_date
1	16073	276	1	860	16.49	2023-11-17 18:02:24.170979 +00:00
2	16125	304	1	135	16.49	2023-11-17 18:02:24.170979 +00:00
3	16160	318	1	224	14.99	2023-11-17 18:02:24.170979 +00:00
4	16161	319	1	15	14.99	2023-11-17 18:02:24.170979 +00:00
5	16358	433	1	691	16.49	2023-11-17 18:02:24.170979 +00:00
6	16373	439	1	786	14.99	2023-11-17 18:02:24.170979 +00:00
7	16449	480	2	822	14.99	2023-11-17 18:02:24.170979 +00:00
8	16618	571	1	228	14.99	2023-11-17 18:02:24.170979 +00:00
9	16702	14	1	346	14.99	2023-11-17 18:02:24.170979 +00:00
10	16717	19	2	110	14.99	2023-11-17 18:02:24.170979 +00:00
11	16790	53	2	856	14.99	2023-11-17 18:02:24.170979 +00:00
12	16796	55	1	1027	14.99	2023-11-17 18:02:24.170979 +00:00
13	16802	57	2	152	14.99	2023-11-17 18:02:24.170979 +00:00
14	16822	67	2	331	14.99	2023-11-17 18:02:24.170979 +00:00
15	16828	71	1	272	14.99	2023-11-17 18:02:24.170979 +00:00
16	16856	85	1	690	14.99	2023-11-17 18:02:24.170979 +00:00
17	16877	101	1	468	14.99	2023-11-17 18:02:24.170979 +00:00
18	16882	103	1	658	14.99	2023-11-17 18:02:24.170979 +00:00
19	16883	104	1	163	16.49	2023-11-17 18:02:24.170979 +00:00
20	16960	142	1	575	14.99	2023-11-17 18:02:24.170979 +00:00

## Exercice 18

Un nouveau client possédant les informations suivantes souhaite louer des films :

M. Guillaume Ransome

Adresse : Rue du centre, 1260 Nyon

Pays : Suisse / Switzerland

Téléphone : 021/360.00.00

E-mail : [gr@bluewin.ch](mailto:gr@bluewin.ch)

Ce client est rattaché au magasin 1.

Insérez-le dans la base de données, avec toutes les informations requises pour lui permettre de louer des films.

1. Spécifiez les attributs (colonnes) lors de l'insertion. *Indication : plusieurs requêtes sont nécessaires.*
2. Pour chaque nouveau tuple, la base de données doit générer l'id. Pourquoi ne pouvez-vous pas le faire ?
3. Pour chaque clé étrangère pour laquelle une valeur est requise, une requête doit donner cette valeur. On considère que l'ensemble des requêtes nécessaires sera fait dans une transaction, ainsi, seules vos modifications de la base de données seront effectives (pas de soucis de concurrence avec une éventuelle autre application).
4. Ecrivez une requête d'interrogation, qui montrera que l'ensemble des opérations s'est bien déroulé.

*Indication : **Guillaume Ransome** est un identifiant unique suffisant.*

## Requête

```
INSERT INTO
    city (city, country_id)
VALUES ('Nyon', (
    SELECT
        country_id AS C
    FROM country
    WHERE country='Switzerland'))
RETURNING *;

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','')
RETURNING *;

INSERT INTO
```



```

        customer
(store_id,first_name,last_name,email,address_id,active,create_date)
VALUES (1,'Guillaume','Ransome','gr@bluewin.ch',(
        SELECT
            max(address_id)
        FROM address),true, now())
RETURNING *;

```

## Résultats

	city_id	city	country_id	last_update
1	601	Nyon	91	2023-11-17 18:04:37.134099 +00:00

	address_id	address	address2	district	city_id	postal_code	phone	last_update
1	606	Rue du centre	<null>		601	1260	022 360 00 00	2023-11-17 18:05:39.055677 +00:00

	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	600	1	Guillaume	Ransome	gr@bluewi...	606	• true	2023-11-17 18:0...	2023-11-17 18:0...

## Réponse

Nyon n'existe pas, il faut donc la créer.

## Requête

```

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 = 'Guillaume' AND last_name = 'Ransome';

```

## Résultats (total: 1)

	first_name	last_name	address	postal_code	city	phone	country	email	store_id
1	Guillaume	Ransome	Rue du centre	1260	Nyon	022 360 00 00	Switzerland	gr@bluewin.ch	1