Nom des pays dont l'espérance de vie est supérieure à 70 ans

```
SELECT nom_pays FROM country
WHERE lifeexpectancy>70;
```

$\Pi_{nom\_pays}(\sigma_{lifeexpectancy>70}(country))$

Nom des capitales des pays du continent '`Europe`' [1]

```
SELECT name_city FROM city
INNER JOIN country ON
 city.id=country.capital
 AND country.continent='Europe';
```

$\Pi_{name\_city}(city \underset{city.id=country.capital \wedge country.continent='Europe'}{\bowtie} country)$

Régions dans lesquelles tous les pays ont une espérance de vie supérieure à 80 ans

```
SELECT region FROM country AS c1
WHERE NOT IN(
 SELECT DISTINCT region FROM country AS c2 WHERE
 c2.lifeexpectancy<=80
);
```

Nom des pays dans lesquels il existe deux villes différentes ayant exactement le même nombre d'habitants

```
SELECT countryname FROM country
INNER JOIN city AS c1 ON
 c1.countrycode=country.countrycode
INNER JOIN city AS c2 ON
 c2.countrycode=c1.countrycode
 AND c2.population_city=c1.population_city
 AND NOT c2.id=c1.id;
```

Nom des pays dont le nombre d'habitants est supérieur à 20 millions [2]

```
SELECT name_country FROM country
WHERE population_country>20000000;
```

$\Pi_{name\_country}(\sigma_{population\_country>20000000}(country))$

Nom des capitales des pays dont l'espérance de vie est inférieure à 60 ans

```
SELECT name_city FROM city
INNER JOIN country ON
 city.id=country.capital
 AND country.lifeexpectancy<60;
```

$\Pi name\_city(city \underset{city.id=country.capital \wedge lifeexpectancy<60}{\bowtie} country)$

Districts de France dans lesquels toutes les villes ont une population supérieure à 500 habitants

```
SELECT district FROM city AS c1
WHERE NOT IN(
 SELECT DISTINCT district FROM city AS c2
 INNER JOIN country ON
 c2.population_city<=500
 AND name_country='France'
 AND country.countrycode=c2.countrycode
);
```

Nom des pays dans lesquels la capitale est la ville la plus peuplée

```
SELECT name_country FROM country
WHERE NOT EXISTS(
 SELECT * FROM city AS c1
 WHERE c1.id=capital
 INNER JOIN city AS c2 ON
 c2.population_city>c1.population_city
 AND NOT c2.id=c1.id
);
```

Nom des capitales des pays du continent '`Asia`' avec moins de 10 millions d'habitants [3]

```
SELECT name_city FROM city
INNER JOIN country ON
 continent='Asia'
 AND population_city<10000000;
 AND id=capital
```

$$\Pi_{name\_city}(city \underset{continent='Asia' \wedge population\_city>10000000 \wedge id=capital}{\bowtie} country)$$

Régions dans lesquelles aucun pays n'a un nom qui commence par la lettre 'P'

```
SELECT region FROM country AS c1
WHERE NOT EXISTS(
 SELECT DISTINCT region FROM country AS c2
 WHERE LEFT(c2.name_country, 1)='P'
);
```

Régions dans lesquels tous les pays ont une capitale avec plus de 100 000 habitants

```
SELECT region FROM country AS c1
WHERE NOT IN(
 SELECT DISTINCT region FROM country AS c2
 INNER JOIN city ON
 id=capital
 AND population_city<=100000
);
```

Nom des villes qui sont capitale d'un pays et aussi ville d'un autre pays sur le même continent [4]

```sql
SELECT name_city FROM city AS cit1
INNER JOIN city AS cit2 ON
 cit2.name_city=cit1.name_city
 AND NOT cit2.countrycode=cit1.countrycode
INNER JOIN country AS cou1 ON
 cit1.id=cou1.capital
INNER JOIN country AS cou2 ON
 cit2.countrycode=cou2.countrycode
 AND cou2.continent=cou1.continent;
```

Quelle(s) requête(s) calcule(nt) le nom des villes françaises de plus de 200 000 habitants ? [5]

```sql
SELECT name_city FROM city WHERE
 population_city>200000
 AND countrycode='FRA';
```

⇒ OK

```sql
SELECT A.name_city FROM city AS A, country AS B WHERE
 population_city>200000
 AND B.countrycode='FRA';
```

⇒ On va avoir des problèmes à cause du produit cartésien : certaines lignes pourront avoir `B.countrycode='FRA'` et `A.countrycode='ENG'`

---

1. Dans la requête, pas besoin de `country.countrycode=city.countrycode` car `id` est une clé primaire.↩
2. La méthode adoptée pour la question demandant « le nom des pays dont la population totale est au moins 50 millions » pourrait s'assimiler à du plagiat vis-à-vis de cette question.↩
3. Syntaxe digne d'un collégien… On supposera ici que ce sont les villes qui ont moins de 10 millions d'habitants. L'adaptation de la requête pour l'autre sens possible de la question est néanmoins trivial.↩
4. Une fois de plus, syntaxe vomitive. On supposera qu'on recherche les noms tels qu'ils puissent faire référence à la fois à la capitale d'un pays P1 et à une ville d'un pays P2 (avec P1 $\neq$ P2 et P1.continent=P2.continent)↩
5. On supposera qu'on a `countrycode='FRA'` ⇒ `name_country='France'`↩

Défini°

AVNUM	AVNOM	CAPACITE	LOCALISATION
1	A300	300	NICE
2	A310	300	NICE
3	B707	250	PARIS
4	A300	280	LYON
5	CONCORDE	160	NICE
6	B747	460	PARIS
7	B707	250	PARIS
8	A310	300	TOULOUSE
9	MERCURE	180	LYON
10	CONCORDE	160	PARIS

un **tuple** de la relation AVION = une **ligne** de la table AVION

Une **donnée numérique** de la relation AVION = une **valeur de type « entier »** de la table AVION

un **attribut** de la relation AVION = une **colonne** de la table AVION

L'**extension** dela **relation** AVION = La **table** AVION

Exemple de schéma d'une relation : "Déclaration de type "
Nom de la relation : Exemple : AVION(AVNUM, AVNOM, CAPACITE, LOCALISATION)
Ensemble des attributs : AVNUM : entier
Domaine de chaque attribut : AVNOM, LOCALISATION : chaîne de caractères limitée à 30
Contraintes d'intégrité : CAPACITÉ : entier < 1000

↳ condi° logiq q doit ê satisfaite par les données

clefs prim^r =>

Exemple
Tuple : RESULTAT(INE, SESSION, UE, NOTE)
Comme chaque étudiant ne peut avoir qu'une note par session pour chaque UE, on a comme clef primaire : (INE, SESSION, UE)

↳ instanc ↳ a un dom^n : ens. de valeurs admissibles

Clefs étrangères

Dépendance d'inclusion :

"E inclus dans F" notée R.E ⊆ S.F : les valeurs de l'attribut de R.E DOIVENT être incluse dans les valeurs existantes dans S.F

- quand on insère dans R une nouvelle valeur pour l'attribut E,
- on doit s'assurer que cette valeur existe dans l'attribut F de S

Exemple de clef étrangère :
AVNUM est clef étrangère de VOL car on a la dépendance d'inclusion :

- VOL.AVNUM ⊆ AVION.AVNUM (clef étrangère en partie gauche & clef primaire en partie droite)
- AVNUM est clé primaire de AVION
- pour ajouter un vol dans la relation VOL, l'avion correspondant doit figurer dans la relation AVION.

R

A	B
1	2
1	2
3	4

$\sigma_{B<4}(R)$

A	B
1	2
1	2

$\sigma_{condi°}(R)$: WHERE condi°

projec° $\pi_{attributs}(R)$ => SELECT attributs

R

A	B	C
1	2	4
1	2	6
3	4	7

$\pi_{A,B}(R)$

A	B
1	2
3	4

R

A	B
1	2
3	4
5	6

S

B	C
1	2
3	4

R×S

A	R.B	S.B	C
1	2	1	2
1	2	3	4
3	4	1	2
3	4	3	4
5	6	1	2
5	6	3	4

=> produit cartésiens

<=> SELECT * From R,S

<=> Select * From R Cross Join S

R

A	B
1	2
3	4
5	6

S

B	C
2	2
2	3
4	6

R⋈S

A	B	C
1	2	2
1	2	3
3	4	6

jointure natur^l

↳ R Natural Join S

R

A	B
1	2
3	4
5	6

S

C	D
2	2
2	3
4	6

$R \bowtie_{B \neq C} S$

A	B	C	D
1	2	4	6
3	4	2	2
3	4	2	3
5	6	2	2
5	6	2	3
5	6	4	6

jointure

From R join S on (condi°)

```
SELECT F1.realisateur
FROM Film F1
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                  WHERE NOT EXISTS (SELECT F2.realisateur
                                    FROM Film F2
                                    WHERE F2.titre=S.titre
                                        AND
                                    F1.realisateur=F2.realisateur))
```

union

R

A	B
1	2
3	4
5	6

S

A	B
1	2
7	8

R∪S

A	B
1	2
3	4
5	6
7	8

SELECT * From R
UNION
SELECT * From S

↳ R Natural Join S

R

A	B
1	2
3	4
5	6

S

A	B
1	2
7	8

R–S

A	B
3	4
5	6

=> NOT IN

ex : les réalisateurs dont tous les films passent dans tous les ciné

$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

R÷S

B
b1
b4

pour un â B il doit avoir le contenu S

R

A	B
1	2
3	4
5	6

S

A	B
1	2
7	8

R∩S

A	B
1	2

= intersect

ex nom de pays esp vie >70

```
SELECT name_country
From country
WHERE life_exp > 70
```

$\pi_{name\_country}(\sigma_{life\_exp>70}(country))$

ex nom des capitales des pays d'europe

```
SELECT name_city From city
INNER JOIN country
```