Pour les exemples, on considère une base de données avec 3 tables dont les schémas relationnels sont :

- film (<u>id</u>, titre, annee, directeur, budget, recette)
- acteur (<u>id</u>, nom)
- casting (id_film, id_acteur)

Une **clé** d'une table est un ensemble minimal d'attributs permettant d'identifier de façon unique chaque enregistrement.
La **clé primaire** d'une table est une clé dont on garantit l'unicité même après ajout dans la table.
Une **clé étrangère** est un attribut (ou ensemble d'attributs) faisant référence à une clé primaire d'une autre table.

Syntaxe générale de `SELECT`, dans cet ordre (`[...]` indiquant une commande optionnelle) :

```
SELECT [DISTINCT] expr1 [AS alias1], expr2, ...
FROM table1 [AS alias1], table2, ...
[WHERE condition]
[GROUP BY expr
[HAVING condition]]
[ORDER BY expr [DESC]]
[LIMIT entier
[OFFSET entier]]
```

- `SELECT [DISTINCT] expr1 [AS alias1], expr2, ...`
  Renvoie une table dont les colonnes correspondent à `expr1`, `expr2`...
  `expr1`, `expr2`... sont des expressions, pouvant contenir des attributs, calculs et fonctions. Si un attribut `attr` est ambigu (car il est le même dans 2 tables `t1` et `t2`), il faut le préfixer par son nom de table, par ex. `t1.attr`.
  $*$ est un raccourci pour selectionner toutes les colonnes.
  `AS` renomme une colonne pour, par exemple, y faire référence ensuite.
  `DISTINCT` supprime les doublons.
  *Obtenir tous les acteurs (sans doublon)* : `SELECT DISTINCT nom FROM acteur;`
  *Films avec leur profit* : `SELECT titre, recette - budget AS profit FROM film;`
- `FROM table1 [AS alias1], table2, ...`
  Tables d'où les valeurs sont sélectionnées.
  `table1, table2` est la table correspondant au produit cartésien de `table1` et `table2`.
  `table1 JOIN table2 ON colonne1 = colonne2` réalise la jointure de `table1` et `table2`, où la `colonne1` de `table1` est identifiée avec `colonne2` de `table2`. On peut mettre plusieurs `JOIN` à la suite :
  *Tous les directeurs et acteurs ayant travaillé ensemble* :
  ```
  SELECT directeur, nom
  FROM film JOIN casting ON film.id = id_film
   JOIN acteur ON id_acteur = acteur.id;
  ```
- `[WHERE condition]`
  Ne considère que les enregistrements vérifiant `condition`. `condition` peut contenir des attributs, calculs, `AND`, `OR`, `<`, `<=`, `!=`, `LIKE`, `IN`...
  *Tous les directeurs qui sont aussi acteurs* :
  `SELECT DISTINCT directeur FROM film, acteur WHERE directeur = nom;`
- `[GROUP BY expr`
  `[HAVING condition]]`
  Regroupe tous les enregistrements ayant la même valeur `expr` en un seul enregistrement. Seuls les groupes vérifiant `condition` sont renvoyés.
  Les fonctions d'agrégations (dans un `SELECT` ou `HAVING`) s'appliquent alors pour chaque groupe : `COUNT(attribut)` (nombre d'enregistrements non `NULL`), `COUNT(*)` (nombre d'enregistrements), `SUM(attribut)`, `MAX(attribut)`, `AVG(attribut)` (moyenne), ...
  *Nombre de films réalisés chaque année depuis 2000* :
  `SELECT annee, COUNT(*) FROM film WHERE annee >= 2000 GROUP BY annee;`
  *Directeurs ayant rapporté au moins 1 milliard* :
  `SELECT directeur FROM film GROUP BY directeur HAVING SUM(recette) >= 1000000000;`
- `[ORDER BY expr [DESC]]`
  Trie les enregistrements selon `expr`, croissant par défaut (décroissant si `DESC` est utilisé).

*Acteurs triés par le nombre de films joués* :

```
SELECT nom, COUNT(*) AS nb_films
FROM acteur JOIN casting ON acteur.id = id_acteur
 JOIN film ON film.id = id_film
GROUP BY nom
ORDER BY nb_films DESC;
```

- `[LIMIT n`
  `[OFFSET p]]`
  Affiche seulement les $n$ premiers enregistrements (en commençant à partir du $(p+1)$ème). Souvent utilisé après un `ORDER BY`.
  *Deuxième film à plus gros budget* :

```
SELECT titre FROM film
ORDER BY budget DESC
LIMIT 1 OFFSET 2;
```

- **Sous-requêtes** : il est possible d'utiliser un `SELECT` renvoyant une seule valeur à l'intérieur d'un autre `SELECT`, dans une condition ou un calcul.
  *Tous les acteurs du film à plus gros budget* :

```
SELECT nom FROM acteur
 JOIN casting ON id_acteur = acteur.id
 JOIN film ON id_film = film.id
WHERE titre = (SELECT titre FROM film ORDER BY budget DESC LIMIT 1);
```

- **Opérateurs ensemblistes** :
  Étant donné deux requêtes de la forme `SELECT ...` renvoyant deux relations `table1` et `table2` de **même schéma relationnel**, il est possible d'obtenir leur union, intersection et différence avec `UNION`, `INTERSECT`, `MINUS`.
  *Exemple* :

| table1 | | |
|---|---|---|
| attr1 | attr2 | attr3 |
| a1 | a2 | a3 |
| b1 | b2 | b3 |

| table2 | | |
|---|---|---|
| attr1 | attr2 | attr3 |
| a1 | a2 | a3 |
| c1 | c2 | c3 |

| attr1 | attr2 | attr3 |
|---|---|---|
| a1 | a2 | a3 |
| b1 | b2 | b3 |
| c1 | c2 | c3 |

Résultat de `SELECT * FROM table1 UNION SELECT * FROM table2;`

| attr1 | attr2 | attr3 |
|---|---|---|
| b1 | b2 | b3 |

Résultat de `SELECT * FROM table1 MINUS SELECT * FROM table2;`