Annales Mines-Ponts SQL

2015–2026

Livret de corrigés — requêtes commentées et variantes

MP*1 MP*2 MP

Comment utiliser ces corrigés

Ce livret est le pendant du livret d'énoncés : un corrigé par annale, dans le même ordre. Cherchez d'abord au stylo, sans document, puis venez comparer.

Chaque question donne la requête attendue, et une remarque seulement là où l'énoncé tend un piège. Une **variante** signale une seconde écriture également acceptable : si la vôtre diffère sans être fausse, c'est souvent celle-là que vous avez trouvée. La mention **hors programme** signale une écriture correcte mais non exigible au concours. Les requêtes composées en italique sont celles de l'énoncé, reproduites telles quelles.

Les encadrés « Rapport du jury » citent mot à mot le rapport officiel de l'épreuve : ils disent ce que les correcteurs attendent et ce qui a coûté des points.

Vous pouvez tester vos propres requêtes sur le site de la classe, qui les exécute et les compare au résultat attendu.

Liste des corrigés

CCMP 2015 — Tests de validation d'une imprimante

Corrigé de la Partie III — Base de données

Q8 — *Numéros de série des imprimantes dont `Imoy` est comprise strictement entre deux bornes `Imin` et `Imax`.*

```sql
SELECT nSerie
FROM testfin
WHERE Imoy > Imin AND Imoy < Imax
```

« Strictement » commande les comparateurs `>` et `<` : une imprimante exactement sur une borne est exclue.

> Cette question ainsi que les deux suivantes ont été généralement abordées. Les réponses sont globalement pertinentes. Un certain nombre de candidats confondent les requêtes SQL (« SELECT … FROM… ») avec l'importation des modules python (« from… import… »). Certains élèves sont gênés par le fait d'exprimer la requête en fonction de Imin et Imax sous forme littérale.
>
> Rapport du jury CCMP 2015

Q9 — *Numéros de série, écart type et fichier de mesures des imprimantes dont `Iec` est strictement inférieur à la moyenne de la colonne `Iec`.*

```sql
SELECT nSerie, Iec, fichierMes
FROM testfin
WHERE Iec < (SELECT AVG(Iec) FROM testfin)
```

> De nombreux candidats connaissent la fonction d'agrégation AVG(), cependant peu savent la mettre en œuvre. On rappelle que les fonctions d'agrégation sont des fonctions s'appliquant sur un ensemble d'enregistrements. Deux requêtes imbriquées étaient attendues ici.
>
> Rapport du jury CCMP 2015

Q10 — *À partir de `testfin`, numéro de série et fichier de mesures des imprimantes qui n'ont pas été validées en sortie de production.*

```sql
SELECT nSerie, fichierMes
FROM testfin
EXCEPT
SELECT T.nSerie, T.fichierMes
FROM (testfin AS T JOIN production AS P ON T.nSerie = P.nSerie)
```

« Non validée » n'est pas une propriété de la ligne mais une absence de ligne dans `production` : c'est une anti-jointure, qui s'écrit par différence ensembliste et qu'aucun `WHERE` posé sur la table jointe ne remplace.

> Dans cette question, l'objet était de récupérer les données d'une table qui ne figurent pas dans une autre. Beaucoup de candidats ont bien compris qu'il fallait faire une différence, mais la structure globale de la requête proposée est généralement incorrecte. Les candidats ne semblent pas maîtriser le résultat renvoyé par une jointure.
>
> Rapport du jury CCMP 2015

Variante — avec une sous-requête et `NOT IN` — hors programme

```sql
SELECT nSerie, fichierMes
FROM testfin
WHERE nSerie NOT IN (SELECT nSerie
                     FROM production)
```

CCMP 2016 — Modélisation de la propagation d'une épidémie

Corrigé de la Partie I — Tri et bases de données

> La maîtrise de la syntaxe de base des langages python et SQL est absolument indispensable. Le respect de l'indentation est capital, et la lisibilité de l'écriture est appréciée : il est rappelé que le doute ne bénéficie pas souvent au candidat.
>
> Rapport du jury CCMP 2016

Q5 — *Parmi `nom`, `iso`, `annee`, quels attributs peuvent servir de clé primaire ? Un couple le peut-il ?*
Aucun des trois pris isolément : `nom` et `iso` se répètent, un même pays étant suivi sur plusieurs années, et `annee` se répète d'un pays à l'autre. En revanche le couple `(iso, annee)` convient, un pays n'ayant qu'une ligne par année. Le couple `(nom, annee)` convient sur l'extrait présenté, mais seul `iso` est déclaré unique par l'énoncé : c'est donc `(iso, annee)` que l'on retient.

> La notion de clef primaire est très mal maîtrisée. On a ainsi pu lire des phrases très étonnantes, comme « l'attribut iso peut servir de clef primaire, et de même le couple iso/année peut servir de clef primaire car elle renvoie à une seule ligne du tableau ». Un travail supplémentaire sur cette notion importante semble nécessaire.
>
> Rapport du jury CCMP 2016

Q6 — *Données de l'année 2010 pour les pays où le nombre de décès dus au paludisme est* ≥ 1000.

```sql
SELECT *
FROM palu
WHERE annee = 2010 AND deces >= 1000
```

> Une requête commençant par FROM palu IMPORT... laisse dubitatif sur la qualité du travail de préparation des candidats sur le langage SQL. La majorité des candidats a cependant traité correctement cette question. Il est aussi rappelé à certains candidats créatifs que l'épreuve d'informatique n'est pas une épreuve d'anglais, et que bricoler une instruction vague avec des pseudo-mots d'anglais n'est pas suffisant pour écrire une requête en SQL.
>
> Rapport du jury CCMP 2016

Q7 — *Taux d'incidence du paludisme en 2011 pour les pays de `palu` (nouveaux cas pour 100 000 habitants).*

```sql
SELECT P.nom, 100000.0 * P.cas / D.pop AS incidence
FROM (palu AS P JOIN demographie AS D ON P.iso = D.pays AND P.annee = D.periode)
WHERE P.annee = 2011
```

> Cette requête comportait plusieurs difficultés. Tous les candidats n'ont pas perçu la nécessité d'une jointure symétrique, qui fut par ailleurs souvent mal écrite. Il fallait noter que la condition de jointure faisait intervenir deux attributs pour chaque table.
>
> Rapport du jury CCMP 2016

Q8 — *Nom du pays ayant eu le deuxième plus grand nombre de nouveaux cas en 2010 (pas d'ex æquo).*

```sql
SELECT nom
FROM palu
WHERE annee = 2010
ORDER BY cas DESC
LIMIT 1 OFFSET 1
```

`LIMIT` et `OFFSET` ne désignent un rang que si un `ORDER BY` le définit : sans lui, la « deuxième » ligne est celle que le moteur produit en second, sans aucune garantie.

> Question plus difficile, que des candidats malins ont traité astucieusement avec LIMIT et OFFSET, profitant d'une certaines ambiguïté du programme sur ce qui était exigible. On pouvait aussi s'en sortir avec des sous-requêtes.
>
> Rapport du jury CCMP 2016

Variante — sans `LIMIT`, avec deux sous-requêtes scalaires

```sql
SELECT nom
FROM palu
WHERE annee = 2010 AND cas = (SELECT MAX(cas)
                              FROM palu
                              WHERE annee = 2010 AND cas < (SELECT MAX(cas)
                                                            FROM palu
                                                            WHERE annee = 2010))
```

CCMP 2017 — Étude de trafic routier

Corrigé de la Partie VI — Base de données

> Les clefs pour obtenir une note honorable à cette épreuve sont de bien maîtriser la syntaxe de base de Python et du SQL de s'assurer que les programmes font exactement - et pas « à peu près » - ce qui est demandé par l'énoncé, et de proposer des programmes concis et bien présentés/indentés.
>
> Rapport du jury CCMP 2017

Q26 — *Identifiants des croisements atteignables en utilisant une seule voie à partir du croisement d'identifiant* `c`*.*

```sql
SELECT id_croisement_fin
FROM Voie
WHERE id_croisement_debut = c
```

`Voie` contient déjà les deux identifiants, et l'orientation compte : une voie qui **arrive** en `c` n'en part pas.

> Cette requête simple, qui ne nécessitait pas de jointure, a été relativement mal traitée.
>
> Rapport du jury CCMP 2017

Q27 — *Longitudes et latitudes des croisements atteignables en utilisant une seule voie à partir du croisement* `c`*.*

```sql
SELECT C.longitude, C.latitude
FROM (Voie AS V JOIN Croisement AS C ON V.id_croisement_fin = C.id)
WHERE V.id_croisement_debut = c
```

> Les candidats semblent maîtriser plutôt mieux que l'an dernier la syntaxe d'une jointure, mais ne maîtrisent pas encore assez la notion : combien de fois a-t-on vu ON Croisement.id=Voie.id dans la condition de jointure ?
>
> Rapport du jury CCMP 2017

Variante — en nommant aussi le croisement de départ

```sql
SELECT C2.longitude, C2.latitude
FROM ((Croisement AS C1 JOIN Voie AS V ON C1.id = V.id_croisement_debut)
      JOIN Croisement AS C2 ON V.id_croisement_fin = C2.id)
WHERE C1.id = c
```

Q28 — *Que renvoie la requête suivante ?*

```sql
SELECT V2.id_croisement_fin
FROM   Voie as V1
JOIN   Voie as V2
ON     V1.id_croisement_fin = V2.id_croisement_debut
WHERE  V1.id_croisement_debut = c
```

La table `Voie` est jointe à elle-même : `V1` part de `c`, et `V2` repart du point d'arrivée de `V1`. La requête renvoie donc les croisements atteignables en **exactement** deux voies depuis `c`, et non en au plus deux. Le croisement `c` peut y figurer, un chemin de longueur 2 pouvant revenir à son départ (aller-retour par les deux voies d'une route à double sens). Faute de `DISTINCT`, un croisement atteint par deux chemins distincts apparaîtrait deux fois.

> Question assez peu traitée.
>
> Rapport du jury CCMP 2017

CCMP 2018 — Mesures de houle

Corrigé de la Partie IV — Base de données relationnelle

Q19 (a) — *Numéro d'identification et nom de site des bouées localisées en Méditerranée.*

```sql
SELECT idBouee, nomSite
FROM Bouee
WHERE localisation = 'Mediterranee'
```

La valeur cherchée doit être écrite exactement comme elle est stockée : `'Mediterranee'`, sans accent.

Q19 (b) — *Numéro d'identification des bouées où il n'y a pas eu de tempêtes.*

```sql
SELECT idBouee
FROM Bouee
EXCEPT
SELECT idBouee
FROM Tempete
```

Une absence de ligne ne se filtre pas par un `WHERE` : cette anti-jointure s'écrit comme une différence ensembliste.

Variante — avec une sous-requête et `NOT IN` — hors programme

```sql
SELECT idBouee
FROM Bouee
WHERE idBouee NOT IN (SELECT idBouee
                      FROM Tempete)
```

Q19 (c) — *Pour chaque site, la hauteur maximale enregistrée lors d'une tempête.*

```sql
SELECT B.nomSite, MAX(T.Hmax)
FROM (Bouee AS B JOIN Tempete AS T ON B.idBouee = T.idBouee)
GROUP BY B.nomSite
```

Le regroupement porte sur le site (`nomSite`) et non sur la bouée : un même site peut porter plusieurs bouées.

> La première requête SQL ne pose pas de problème, sauf à ceux qui ne se sont manifestement pas entraînés sur ce langage. Les autres requêtes, plus complexes, ont eu des succès variables. Parmi les erreurs rencontrées, l'appel à un attribut via attribut.table à la place de table.attribut . De plus, la condition de jointure après ON est bien une condition, et non seulement le nom d'un attribut. Rappelons que la seule syntaxe exigible du programme pour effectuer une jointure est table1 JOIN table2 ON condition (jointure symétrique simple). Trop de candidats ont voulu se lancer dans des NATURAL JOIN et autres FULL JOIN sans maîtriser précisément leurs effets.
>
> Rapport du jury CCMP 2018

CCMP 2019 — Autour des nombres premiers

Corrigé de la Partie IV — Analyse de performance de code

Q25 — *Pourquoi ne peut-on pas utiliser `nom` comme clé primaire de la table `fonctions` ?*
Une clé primaire doit identifier chaque ligne de façon unique. Or une ligne de `fonctions` décrit un **test**, et `nom` y désigne la fonction testée : la même fonction est évaluée par plusieurs algorithmes et sur plusieurs machines, donc sa valeur se répète (`li` apparaît trois fois dans l'extrait de l'énoncé). C'est `id`, qui change à chaque test, qui joue ce rôle.

> L'erreur la plus courante a été de dire que les deux tables possédant un attribut de même nom, celui-ci ne pouvait servir de clé primaire pour l'une des tables : cela n'a rien à voir avec la définition d'une clé primaire.
>
> Rapport du jury CCMP 2019

Q26 (1) — *Nombre d'ordinateurs disponibles et leur quantité moyenne de mémoire vive.*

```sql
SELECT COUNT(*), AVG(ram)
FROM ordinateurs
```

Q26 (2) — *Noms des PC sur lesquels l'algorithme `rectangles` n'a pas été testé pour la fonction `li`.*

```sql
SELECT nom
FROM ordinateurs
EXCEPT
SELECT teste_sur
FROM fonctions
WHERE nom = 'li' AND algorithme = 'rectangles'
```

Cette anti-jointure porte un piège d'homonymie : `nom = 'li'` désigne `fonctions.nom`, le nom de la **fonction**, et non `ordinateurs.nom`.

> Peu de candidats ont perçu la nécessité d'une sous-requête (ou d'un EXCEPT) pour la deuxième requête. En revanche, la syntaxe d'une jointure semble mieux maîtrisée que les années précédentes.
>
> Rapport du jury CCMP 2019

Variante — avec une sous-requête et `NOT IN` — hors programme

```sql
SELECT nom
FROM ordinateurs
WHERE nom NOT IN (SELECT teste_sur
                  FROM fonctions
                  WHERE nom = 'li' AND algorithme = 'rectangles')
```

Q26 (3) — *Pour la fonction `Ei`, trier les tests du plus lent au plus rapide ; pour chaque test : algorithme, PC, et puissance du PC.*

```sql
SELECT F.algorithme, O.nom, O.gflops
FROM (ordinateurs AS O JOIN fonctions AS F ON O.nom = F.teste_sur)
WHERE F.nom = 'Ei'
ORDER BY F.temps_exec DESC
```

« Du plus lent au plus rapide » fait partie de la réponse : `ORDER BY F.temps_exec DESC`, sur une colonne qui n'a pas à figurer dans le `SELECT`.

CCMP 2020 — Images de vagues et de structures

Corrigé de la Partie I, § I.a — Chargement d'un modèle 3D

Q1 — *Compter le nombre de maillages que contient le modèle du bateau.*

```sql
SELECT COUNT(*)
FROM maillages_bateau
```

> Trop de candidats ne maitrisent pas la syntaxe d'une fonction d'agrégation, confondent COUNT avec SUM, ou n'utilisent pas la bonne relation.
>
> Rapport du jury CCMP 2020

Q2 — *Numéros des facettes (`numero`) du maillage nommé « gouvernail ».*

```sql
SELECT F.numero
FROM (maillages_bateau AS M JOIN faces AS F ON M.id = F.maillage)
WHERE M.nom = 'gouvernail'
```

> Mieux réussie que Q1, sans doute parce que l'énoncé donnait une exemple de jointure à la question suivante.
>
> Rapport du jury CCMP 2020

Variante — avec une sous-requête scalaire — suppose qu'un seul maillage porte ce nom

```sql
SELECT numero
FROM faces
WHERE maillage = (SELECT id
                  FROM maillages_bateau
                  WHERE nom = 'gouvernail')
```

Q3 — *Qu'exprime la requête suivante ?*

```sql
-- Requête donnée par l'énoncé, forme d'origine conservée (double espace et guillemets compris).
SELECT (MAX(x)-MIN(x))
FROM sommets AS s JOIN faces AS f JOIN  maillages_bateau AS m
ON (s.id=f.s1 OR s.id=f.s2 OR s.id=f.s3) AND f.maillage=m.id
WHERE m.nom="coque"
```

La requête renvoie l'étendue de la coque selon l'axe x : la différence entre la plus grande et la plus petite abscisse des sommets appartenant à la coque, c'est-à-dire la dimension de sa boîte englobante le long de x. Le triple `OR` rattache à chaque facette ses trois sommets ; un sommet partagé par plusieurs facettes est alors compté plusieurs fois, ce qui est sans effet sur `MAX` et `MIN` mais fausserait un `COUNT` ou une `SUM`.

> Assez bien réussie, mais il manque souvent une information : selon l'axe x, ou la mention d'une largeur/longueur/taille.
>
> Rapport du jury CCMP 2020

Variante — la même requête, écrite dans la forme conseillée — le `OR` dans le `ON`, imposé par l'énoncé, sort du cadre du programme (conjonction d'égalités)

```sql
SELECT MAX(S.x) - MIN(S.x)
FROM ((faces AS F JOIN maillages_bateau AS M ON F.maillage = M.id)
      JOIN sommets AS S ON S.id = F.s1 OR S.id = F.s2 OR S.id = F.s3)
WHERE M.nom = 'coque'
```

CCMP 2021 — Marchons, marchons, marchons...

Corrigé de la Partie I — Randonnée

Q1 — *Compter le nombre de participants nés entre 1999 et 2003 inclus.*

```sql
SELECT COUNT(*)
FROM Participant
WHERE ne >= 1999 AND ne <= 2003
```

« Inclus » impose les comparaisons larges `>=` et `<=`, réunies par un `AND` : un encadrement ne s'écrit pas d'un trait.

> Trop de candidats oublient le SELECT quand ils utilisent une fonction d'agrégation. De plus, la syntaxe 1998 < ne < 2004 ne permet pas de répondre à la question ; on préférera écrire par exemple ne > 1998 AND ne < 2004, ce qui n'est pas équivalent en SQL.
>
> Rapport du jury CCMP 2021

Q2 — *Calculer la durée moyenne des randonnées pour chaque niveau de difficulté.*

```sql
SELECT diff, AVG(duree)
FROM Rando
GROUP BY diff
```

> La fonction d'agrégation AVG n'est pas toujours maîtrisée, ainsi que l'usage de GROUP BY.
>
> Rapport du jury CCMP 2021

Q3 — *Extraire le nom des participants pour lesquels la randonnée n°42 est trop difficile.*

```sql
SELECT pnom
FROM Participant
WHERE diff_max < (SELECT diff FROM Rando WHERE rid = 42)
```

> Beaucoup de candidats ont voulu utiliser une jointure entre les tables Rando et Participant en utilisant une condition du type ON rid = pid, montrant ainsi un manque de compréhension de cette notion. Un produit cartésien (suivi d'une clause WHERE) ou l'utilisation d'une sous-requête permettaient de répondre à la question.
>
> Rapport du jury CCMP 2021

Variante — avec un produit cartésien

```sql
SELECT P.pnom
FROM Participant AS P, Rando AS R
WHERE R.rid = 42 AND P.diff_max < R.diff
```

Q4 — *Extraire les clés primaires des randonnées qui ont un ou des homonymes (nom identique et clé primaire distincte), sans redondance.*

```sql
SELECT DISTINCT R1.rid
FROM (Rando AS R1 JOIN Rando AS R2 ON R1.rnom = R2.rnom)
WHERE R1.rid <> R2.rid
```

« Sans redondance » impose le `DISTINCT` ; l'égalité d'appariement `R1.rnom = R2.rnom` va dans le `ON`, la condition `R1.rid <> R2.rid`, qui n'est pas une égalité, dans le `WHERE`.

> Cette question était plus délicate. Une auto-jointure permettait par exemple de répondre à la question. Certains candidats ont également cherché à utiliser une sous-requête ; attention dans ce cas à la confusion entre HAVING et WHERE. De manière générale, beaucoup de candidats utilisent une syntaxe erronée de la forme attribut.table en lieu et place de table.attribut.
>
> Rapport du jury CCMP 2021

Variante — avec une sous-requête et `IN` — hors programme

```sql
SELECT rid
FROM Rando
WHERE rnom IN (SELECT rnom
               FROM Rando
               GROUP BY rnom
               HAVING COUNT(*) > 1)
```

CCMP 2022 — Modélisation numérique d'un matériau magnétique

Corrigé de la Partie II — Recherche dans une base de données de matériaux magnétiques

> Certaines questions étaient d'un niveau élémentaire (importation de modules, requête SQL simple, calcul d'une moyenne...), quand d'autres exigeaient une maîtrise et une compréhension plus fines.
>
> Rapport du jury CCMP 2022

Q6 — *Nom des matériaux dont la température de Curie est strictement inférieure à 500 K.*

```sql
SELECT nom
FROM materiaux
WHERE t_curie < 500
```

Q7 — *Noms des fournisseurs proposant du nickel et prix proposé par chacun pour 4,5 kg.*

```sql
SELECT F.nom_fournisseur, P.prix_kg * 4.5
FROM (fournisseurs AS F JOIN prix AS P ON F.id_fournisseur = P.id_four)
WHERE P.id_mat = 8713
```

> Attention à l'oubli du coefficient 4.5. Certains candidats ne savent pas comment obtenir deux attributs dans la même requête SQL. La syntaxe d'une jointure (et le choix des attributs adéquats) n'est pas assez maîtrisée.
>
> Rapport du jury CCMP 2022

Q8 — *Nom du fournisseur de nickel le moins cher et prix à payer pour 4,5 kg, tous les fournisseurs étant attendus en cas d'égalité.*

```sql
SELECT F.nom_fournisseur, P.prix_kg * 4.5
FROM (fournisseurs AS F JOIN prix AS P ON F.id_fournisseur = P.id_four)
WHERE P.id_mat = 8713
  AND P.prix_kg = (SELECT MIN(prix_kg)
                   FROM prix
                   WHERE id_mat = 8713)
```

Plusieurs fournisseurs peuvent être ex æquo au prix minimal : un `ORDER BY … LIMIT 1` n'en garderait qu'un, d'où la comparaison à la sous-requête `MIN`.

> L'utilisation de MIN a souvent donné lieu à des erreurs de syntaxe.
>
> Rapport du jury CCMP 2022

Q9 — *Nom des matériaux et prix moyen au kilogramme, tous fournisseurs confondus, en se limitant aux moyennes strictement inférieures à 50 €/kg.*

```sql
SELECT M.nom, AVG(P.prix_kg)
FROM (materiaux AS M JOIN prix AS P ON M.id_materiau = P.id_mat)
GROUP BY M.id_materiau, M.nom
HAVING AVG(P.prix_kg) < 50
```

> L'utilisation de GROUP BY et de HAVING (souvent confondu avec WHERE) est souvent erronée.
>
> Rapport du jury CCMP 2022

CCMP 2023 — La typographie informatisée

Corrigé de la Partie II — Gestion de polices de caractères vectorielles

> Certaines questions étaient d'un niveau élémentaire (requête SQL simple, manipulations classiques de listes...), quand d'autres (essentiellement en fin de sujet) exigeaient une maîtrise et une compréhension plus fines.
>
> Rapport du jury CCMP 2023

Q3 — *Compter le nombre de glyphes en roman.*

```sql
SELECT COUNT(*)
FROM Glyphe
WHERE groman = TRUE
```

Le programme ignore le type booléen ; on suit l'énoncé, qui déclare `groman` booléen, avec `groman = TRUE` (les moteurs qui codent les booléens par 0 et 1 acceptent aussi `groman = 1`).

> La syntaxe exacte de COUNT n'est pas toujours maîtrisée : il faut préciser l'attribut, ou utiliser à bon escient la syntaxe COUNT(*).
>
> Rapport du jury CCMP 2023

Q4 — *Description vectorielle du caractère « A » dans la police « Helvetica » en italique.*

```sql
SELECT G.gdesc
FROM ((Police AS P JOIN Glyphe AS G ON P.pid = G.pid)
      JOIN Caractere AS C ON G.code = C.code)
WHERE C.car = 'A' AND P.pnom = 'Helvetica' AND G.groman = FALSE
```

> La syntaxe des jointures n'est pas toujours maîtrisée : le programme officiel de CPGE est très clair à ce sujet. De nombreuses erreurs de syntaxe ont été remarquées, relatives à l'interversion table / attribut : TABLE.ATTRIBUT est la syntaxe correcte, et non pas ATTRIBUT.TABLE.
>
> Rapport du jury CCMP 2023

Q5 — *Noms des familles qui disposent de polices et leur nombre de polices, classés par ordre alphabétique.*

```sql
SELECT F.fnom, COUNT(*) AS nb_polices
FROM (Famille AS F JOIN Police AS P ON F.fid = P.fid)
GROUP BY F.fnom
ORDER BY F.fnom ASC
```

L'ordre des groupes n'est jamais garanti : le classement alphabétique exigé par l'énoncé impose un `ORDER BY F.fnom` explicite.

> La gestion des fonctions d'agrégation a posé problème : il fallait utiliser un GROUP BY, et la syntaxe du tri par ordre alphabétique à l'aide d'un ORDER BY n'est souvent pas maîtrisée.
>
> Rapport du jury CCMP 2023

Variante — en regroupant par identifiant de famille

```sql
SELECT F.fnom, COUNT(*) AS nb_polices
FROM (Famille AS F JOIN Police AS P ON F.fid = P.fid)
GROUP BY F.fid, F.fnom
ORDER BY F.fnom ASC
```

CCMP 2024 — Introduction à deux problèmes en communication numérique

Corrigé de la Partie 1.2 — Exploitation d'analyses existantes

> Certaines questions étaient d'un niveau élémentaire (requête SQL simple, recherche du maximum d'une liste ...), quand d'autres (essentiellement en fin de sujet) exigeaient une maîtrise et une compréhension plus fines.
>
> Rapport du jury CCMP 2024

Q8 — *Liste sans doublon des auteurs présents dans la table* `corpus`*.*

```sql
SELECT DISTINCT auteur
FROM corpus
```

> Question bien traitée dans l'ensemble malgré une syntaxe approximative lors de l'utilisation de DISTINCT. De rares candidats ne connaissent pas la syntaxe SELECT ... FROM ... ; certains proposent des requêtes se terminant par WHERE sans condition. Plusieurs candidats, à cause d'une mauvaise lecture du sujet, pensent que table fait partie du nom de chaque table.
>
> Rapport du jury CCMP 2024

Q9 — *Fréquence d'occurrences (comprise entre 0 et 1) de chaque caractère en « Français » : une colonne avec le symbole, une avec le rapport entre le nombre total d'occurrences du caractère et le nombre total de caractères des textes de tout le corpus.*
L'énoncé est ambigu — une fréquence « en Français », mais un dénominateur « de tout le corpus » — et l'on retient d'abord la lecture cohérente avec « en Français » : occurrences et total de caractères comptés sur les seuls textes français.

```sql
SELECT C.symbole,
       1.0 * SUM(O.nombreOccurrences)
         / (SELECT SUM(nombreCaracteres)
            FROM corpus
            WHERE langue = 'Français')
       AS frequence
FROM ((occurrences AS O JOIN caractere AS C ON O.idCar = C.idCar)
      JOIN corpus AS Co ON O.idLivre = Co.idLivre)
WHERE Co.langue = 'Français'
GROUP BY C.idCar, C.symbole
```

Le facteur `1.0 *` évite la division entière (toutes les fréquences seraient nulles, sans message d'erreur) ; le dénominateur, total des textes français, se calcule à part par une sous-requête scalaire ; on regroupe par la clé `C.idCar`, avec `symbole` pour pouvoir le projeter.

> Requête assez compliquée. Plusieurs candidats proposent des sous-requêtes alors que le sujet demande explicitement UNE requête. L'utilisation des jointures est assez peu maitrisée. Erreurs de syntaxe SQL récurrentes : confusion entre WHERE et HAVING ; confusion entre SUM et COUNT ; utilisation de attribut.table au lieu de table.attribut ; mauvaise maîtrise de GROUP BY.
>
> Rapport du jury CCMP 2024

Variante — sans sous-requête — juste uniquement si chaque caractère a une ligne d'`occurrences` dans chaque livre français

```sql
SELECT C.symbole,
       1.0 * SUM(O.nombreOccurrences) / SUM(Co.nombreCaracteres) AS frequence
FROM ((occurrences AS O JOIN caractere AS C ON O.idCar = C.idCar)
      JOIN corpus AS Co ON O.idLivre = Co.idLivre)
WHERE Co.langue = 'Français'
GROUP BY C.idCar, C.symbole
```

Lecture littérale, à présent : le dénominateur porte sur tout le corpus, ce que seule une sous-requête scalaire peut fournir puisque le `WHERE` écarte le livre anglais du regroupement.

```sql
SELECT C.symbole,
       1.0 * SUM(O.nombreOccurrences)
         / (SELECT SUM(nombreCaracteres)
            FROM corpus)
       AS frequence
FROM ((occurrences AS O JOIN caractere AS C ON O.idCar = C.idCar)
      JOIN corpus AS Co ON O.idLivre = Co.idLivre)
WHERE Co.langue = 'Français'
GROUP BY C.idCar, C.symbole
```

CCMP 2025 — Autour du sac à dos

Corrigé de la Partie 1 — Base de données (fret maritime de conteneurs)

Q1 — *Identifiants des conteneurs et leur ratio tarification/longueur, trié par ordre décroissant, partant de Marseille et allant à Barcelone, avec une mise à disposition avant le 01/01/2025.*

```sql
SELECT idC, 1.0 * val / taille AS ratio
FROM CONTENEURS
WHERE portDepC = 'Marseille' AND portDestC = 'Barcelone' AND dateDisp < '2025-01-01'
ORDER BY 1.0 * val / taille DESC
```

« Avant le 01/01/2025 » exclut ce jour-là : l'inégalité est stricte ; le facteur `1.0 *` force une division réelle, sans quoi `val / taille` vaudrait 0 sur toute la table.

> La syntaxe SQL n'est pas entièrement maitrisée pour une majorité de candidats. Erreurs fréquentes : confusion dans l'ordre des instructions, utilisation de AND (au lieu d'une virgule) entre les différents attributs sélectionnés avec SELECT, utilisation d'une virgule (à la place de AND) entre les conditions spécifiées après WHERE, pas de précision de l'attribut utilisé pour le classement lors de l'utilisation de ORDER BY.
>
> Il est conseillé aux candidats de faire un effort de présentation pour les requêtes SQL : mettre en majuscule les éléments de langage et en minuscule les éléments spécifiques aux tables ; aller à la ligne régulièrement.
>
> Rapport du jury CCMP 2025

Q2 — *Identifiants de navire et leur nombre respectif de conteneurs attribués.*

```sql
SELECT idN, COUNT(*)
FROM CONTENEURS
WHERE idN <> 0
GROUP BY idN
```

La sentinelle `idN = 0` ne désigne pas un navire : on l'écarte par un `WHERE`, donc avant regroupement.

> Idem Q1.
>
> Rapport du jury CCMP 2025

Variante — en passant par une jointure — inutile, aucun attribut de `NAVIRES` n'étant projeté

```sql
SELECT N.idN, COUNT(*)
FROM (NAVIRES AS N JOIN CONTENEURS AS C ON N.idN = C.idN)
GROUP BY N.idN
```

CCMP 2026 — Jeu Quixo à deux joueurs

Corrigé de la Partie 4 — Gestion d'une base de données

Q20 — *Nombre de parties et nombre moyen de tours réalisés par le joueur nommé « John Doe » quand il a commencé la partie (il est le joueur 1).*

```
SELECT COUNT(*), AVG(P.nbtours)
FROM (Partie AS P JOIN Joueur AS J ON P.id_joueur1 = J.id)
WHERE J.nom = 'Doe' AND J.prenom = 'John'
```

Q21 — *Nombre de tours de la partie ayant nécessité le moins de tours, ainsi que les noms et prénoms des joueurs 1 et 2 ayant participé à cette partie (unicité supposée).*

```
SELECT P.nbtours,
       J1.nom AS nom1, J1.prenom AS prenom1, J2.nom AS nom2, J2.prenom AS prenom2
FROM ((Partie AS P JOIN Joueur AS J1 ON P.id_joueur1 = J1.id)
      JOIN Joueur AS J2 ON P.id_joueur2 = J2.id)
WHERE P.nbtours = (SELECT MIN(nbtours) FROM Partie)
```

En cas d'ex æquo, la sous-requête `MIN` renvoie toutes les parties minimales, là où un `ORDER BY … LIMIT 1` n'en garderait qu'une, arbitrairement.

Variante — en triant et en ne gardant que la première ligne

```
SELECT P.nbtours,
       J1.nom AS nom1, J1.prenom AS prenom1, J2.nom AS nom2, J2.prenom AS prenom2
FROM ((Partie AS P JOIN Joueur AS J1 ON P.id_joueur1 = J1.id)
      JOIN Joueur AS J2 ON P.id_joueur2 = J2.id)
ORDER BY P.nbtours ASC
LIMIT 1
```