MP/MP*
PC/PC*
PSI/PSI*
PT/PT*

INFORMATIQUE

1 DEVOIR CORRIGÉ PAR SEMAINE



avec annales
& corrigés commentés

Benjamin Binder

ellipses

MP/MP*
PC/PC*
PSI/PSI*
PT/PT*

Informatique



Benjamin BINDER

ellipses

ISBN 9782340-118935
Dépôt légal : septembre 2026
©Ellipses Édition Marketing S.A.
8/10 rue la Quintinie 75015 Paris



Le Code de la propriété intellectuelle et artistique n'autorisant, aux termes des alinéas 2 et 3 de l'article L. 122-5, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

www.editions-ellipses.fr

Avant-propos

La deuxième année de CPGE comporte **un seul semestre** en informatique. Cela signifie que l'acquisition des connaissances et compétences spécifiées par le programme doit se faire de façon très efficace, dans un temps court. Mis à part les bases de données et le langage SQL, le programme de deuxième année n'est évidemment pas indépendamment de celui de première année : il s'inscrit dans sa continuité. Les corrigés proposés ne font que très peu d'hypothèses sur votre niveau de maîtrise du programme de première année. En ce sens, cet ouvrage a pour objet de vous aider dans votre apprentissage, tout en révisant les notions de base introduites en première année.

Plus précisément, cet ouvrage vous permet de vous **entraîner à un rythme régulier**, pour vous préparer aux devoirs sur table et aux épreuves de concours, écrites notamment. Ainsi, **vingt énoncés** de devoirs relativement courts et ciblés, ainsi qu'**un sujet d'annale**, vous sont proposés, tous accompagnés de leur corrigé. Les **corrigés** vous permettront également de parfaire votre méthode de lecture des énoncés, de les relier aux notions adéquates et de vous organiser pour avoir une note optimale. Pour faciliter l'identification des points importants, les corrigés sont ponctués d'encadrés détaillant les rappels de cours essentiels, les méthodes à connaître et les erreurs à éviter. Les corrigés fournissent donc souvent des réponses bien plus longues et détaillées que ce qui est attendu pour simplement répondre aux questions.

Dans l'optique des devoirs sur table et des écrits des concours, les devoirs de cet ouvrage sont destinés à être résolus en écrivant du code sur papier. Néanmoins, il est tout à fait possible (et même conseillé !), *après* avoir proposé une version sur papier et hors temps de résolution

indiqué, de taper le code dans un IDE (*Integrated Development Environment*) tel que *Spyder* ou *Idle* pour Python, ou dans une interface graphique telle que *DB Browser for SQLite* pour SQL (avec une petite base de test fictive), afin de s'entraîner à utiliser les outils de développement habituels, mais aussi d'en tirer parti pour perfectionner sa maîtrise de la syntaxe des langages en autonomie, et bien sûr pour exécuter et tester le code (avant de regarder les corrigés).

Même s'il peut être frustrant de coder « à l'écrit », il faut en être capable pour les concours. Au-delà de cet aspect pragmatique, l'approche n'est pas si différente de celle que l'on devrait avoir face à un ordinateur, car, même sur machine, il faut être très **rigoureux et méthodique**, en anticipant les erreurs de syntaxe bien sûr, mais aussi les potentielles erreurs d'exécution : il ne suffit pas d'obtenir à la fin un code qui fonctionne, il faut en comprendre chaque ligne écrite et prévoir les résultats, y compris dans des cas limites.

La plupart des signatures des fonctions Python sont données dans les énoncés avec des **annotations de fonction**. Celles-ci indiquent le type attendu des arguments, ainsi que celui de la valeur de retour. Les paramètres sont ainsi suivis du symbole `:` puis du ou des types possibles attendus (séparés par la barre verticale *pipe* `|`), et, de même, la signature de la fonction se termine, entre la parenthèse `)` et les deux points `:`, par le ou les types attendus pour la valeur de retour. Sans rentrer dans les détails, outre les annotations `int` et `float`, `list[T]` désigne une liste dont tous les éléments sont de type `T`, `dict[T1, T2]` désigne un dictionnaire dont les *clés* sont de type `T1` et les *valeurs*, de type `T2`, et `tuple[T1, T2]` désigne un tuple dont le premier élément est de type `T1` et le second, de type `T2`. On peut éventuellement ne pas spécifier les types `T`, `T1` ou `T2`. Ces annotations sont ignorées par l'interpréteur Python, mais elles constituent une bonne pratique pour documenter le code et servent, dans les énoncés, à spécifier précisément ce que l'on attend. Il n'est pas toujours nécessaire, à l'écrit, de reproduire ces annotations dans les réponses ; cela est même contre-indiqué dans les consignes de certains sujets. Ces annotations ne sont pas reportées dans les corrigés de cet ouvrage.

Enfin, dans tous les passages contenant du code (que ce soit en Python ou SQL), le symbole « ↪ » indique simplement, pour des raisons d'espace, la continuation de la ligne précédente (en cohérence avec les numéros de ligne indiqués).

Sommaire

Index thématique 7

Devoirs

- Devoir 1 11
- Devoir 2 15
- Devoir 3 19
- Devoir 4 21
- Devoir 5 23
- Devoir 6 25
- Devoir 7 29
- Devoir 8 33
- Devoir 9 35
- Devoir 10 39
- Devoir 11 43
- Devoir 12 47
- Devoir 13 51
- Devoir 14 55
- Devoir 15 59
- Devoir 16 65
- Devoir 17 69
- Devoir 18 73
- Devoir 19 77
- Devoir 20 81
- Devoir 21 85

Corrigés

- Corrigé du devoir 1 101
- Corrigé du devoir 2 109
- Corrigé du devoir 3 117
- Corrigé du devoir 4 127
- Corrigé du devoir 5 135
- Corrigé du devoir 6 141
- Corrigé du devoir 7 149
- Corrigé du devoir 8 157
- Corrigé du devoir 9 163
- Corrigé du devoir 10 175
- Corrigé du devoir 11 187
- Corrigé du devoir 12 197
- Corrigé du devoir 13 207
- Corrigé du devoir 14 215
- Corrigé du devoir 15 225
- Corrigé du devoir 16 235
- Corrigé du devoir 17 243
- Corrigé du devoir 18 253
- Corrigé du devoir 19 261
- Corrigé du devoir 20 269
- Corrigé du devoir 21 275

Index thématique

Apprentissage
Devoirs 14, 15

Bases de données
Devoirs 1–6

Dictionnaires
Devoirs 7–10, 12

Graphes
Devoirs 11, 16–20

Intelligence artificielle
Devoirs 14–20

Programmation dynamique
Devoirs 8–13

Récursivité
Devoirs 9, 10, 12, 19, 20

Théorie des jeux
Devoirs 16–20

Tri
Devoirs 11, 13, 14

DEVOIRS

Base de données d'un CDI

Thème abordé : bases de données.
Temps de résolution suggéré : 1 heure.

Présentation du devoir

On dispose d'une base de données gérant les emprunts de livres dans le CDI d'un lycée comportant des CPGE. Le schéma relationnel est le suivant :

— **ETUDIANT**(`id_etudiant` INT PK, `nom` TEXT, `prenom` TEXT, `promotion` INT)

— **LIVRE**(`id_livre` INT PK, `titre` TEXT, `auteur` TEXT, `annee_publication` INT, `rayon` TEXT)

— **EMPRUNT**(`id_emprunt` INT PK, `id_etudiant` INT FK, `id_livre` INT FK, `date_emprunt` INT, `date_retour` INT)

Dans ce schéma, `PK` signifie *Primary Key*, c'est-à-dire clé primaire, et `FK` signifie *Foreign Key*, c'est-à-dire clé étrangère.

Contraintes :

— Un livre peut exister en plusieurs exemplaires.

— Un étudiant ne peut emprunter qu'un seul exemplaire d'un même livre à la fois.

— On suppose qu'il n'y a pas d'homonymes parmi les auteurs.

— Tous les livres sont rendus au plus tard à la fin de l'année : la base ne concerne que l'année courante.

— Les dates sont représentées par le numéro du jour de l'année. Une `date_retour` ayant la valeur 0 signifie que le livre n'a pas encore été restitué.

Questions

1. Rappeler brièvement ce que signifient les notions de clés primaire et étrangère.
2. Écrire une requête SQL qui sélectionne les noms et prénoms des étudiants de la promotion 2025.
3. Écrire une requête qui fournit la liste des titres de livres publiés après 2010.
4. Écrire une requête qui compte le nombre total de livres par rayon.
5. Proposer un diagramme entité–association (sans propriétés) correspondant à ce modèle relationnel. Distinguer clairement les types d'entité et d'association et expliquer le rôle de la table `EMPRUNT`.
6. Écrire une requête qui renvoie, pour chaque emprunt en cours, le nom de l'étudiant, son prénom, le titre du livre et la date d'emprunt, triés par dates d'emprunt antéchronologiques.
7. Écrire une requête qui renvoie la liste des auteurs dont au moins deux livres ont été empruntés depuis le début de l'année en cours.
8. Écrire une requête qui, pour chaque étudiant (avec son nom et prénom), calcule le nombre d'emprunts qu'il a réalisés pendant l'année en cours.
9. Que faut-il faire pour que la requête renvoie uniquement les étudiants ayant effectué au moins 5 emprunts ?
10. Écrire une requête qui, pour chaque étudiant de la promotion 2025 ayant effectué au moins 5 emprunts, calcule le nombre d'emprunts du livre qu'il a réalisés pendant l'année en cours.
11. Écrire une requête qui permette de trouver le titre du livre le plus ancien jamais emprunté.
12. Écrire une requête qui renvoie, pour chaque rayon, le délai moyen (en jours) de retour des livres, en n'incluant que les emprunts déjà clôturés.

Mots-clés

Requête SQL ; sélection ; agrégation ; clé primaire ; clé étrangère ; modèle entité–association ; jointures.

Plateforme de cours en ligne

Thème abordé : bases de données.

Temps de résolution suggéré : 1 heure.

Présentation du devoir

On s'intéresse à une plateforme de cours en ligne. Cette plateforme propose des cours suivis par des étudiants, enseignés par des enseignants.

Contraintes :

— Un étudiant peut suivre plusieurs cours (ayant notamment un titre), voire aucun encore.

— On doit connaître la ville de résidence des étudiants.

— Un cours peut être suivi par plusieurs étudiants.

— Chaque cours est assuré par un seul enseignant.

— Un enseignant peut assurer plusieurs cours, mais possède une seule spécialité.

Questions

A. Du modèle Entité–Association au modèle relationnel

1. Identifier les entités et leurs propriétés minimales, au vu des contraintes de l'énoncé.
2. Représenter graphiquement le modèle Entité–Association (sans les propriétés) de la plateforme.
3. Traduire ce modèle en schéma relationnel. Préciser en particulier les clés primaires et étrangères.

B. Requêtes SQL

On ajoute comme contrainte la possibilité pour les étudiants d'avoir des notes pour un cours donné, si bien que l'on considère le schéma relationnel suivant pour la suite :

```
ETUDIANT(idE, nom, ville);
ENSEIGNANT(idEns, nom, specialite);
COURS(idC, titre, idEns);
INSCRIPTION(idE, idC);
NOTE(idE, idC, note);
```

Pour chacune des questions suivantes, répondre par une **requête SQL** donnant :

4. le titre de chaque cours et l'enseignant qui l'assure ;
5. les noms des étudiants inscrits au cours « Bases de données » ;
6. les noms des étudiants ayant (au moins) une note supérieure ou égale à 12 dans le cours « IA ».
7. les moyennes des notes par cours ;
8. les noms des enseignants assurant au moins deux cours ;
9. les noms des étudiants résidant à Paris inscrits à aucun cours ;
10. les noms des étudiants au-dessus de la moyenne générale.

Mots-clés

Requête SQL ; sélection ; agrégation ; modèle entité–association ; jointures ; sous-requêtes ; opérations ensemblistes.

Compétitions sportives

Thème abordé : bases de données.
Temps de résolution suggéré : 1 heure 15.

Présentation du devoir

On s'intéresse à une plateforme de compétitions sportives.

Des joueurs participent à des tournois. Chaque tournoi comporte plusieurs rencontres (matchs). Chaque rencontre oppose exactement deux joueurs et produit un score pour chaque joueur.

On donne le schéma de la base de données relationnelle de cette plateforme :

```
JOUEUR(idJ, nom, pays)
TOURNOI(idT, nomT, ville)
RENCONTRE(idM, idT, idJ1, idJ2, scoreJ1, scoreJ2)
```

Clés étrangères avec leur référence ($\rightarrow$) :

— `RENCONTRE.idT` $\rightarrow$ `TOURNOI.idT`

— `RENCONTRE.idJ1`, `RENCONTRE.idJ2` $\rightarrow$ `JOUEUR.idJ`

Questions

A. Analyse du modèle relationnel

1. Identifier les entités du modèle Entité–Association (EA) correspondant à ce schéma relationnel et leurs propriétés. Commenter la nécessité de la clé primaire `idM`.
2. En identifiant des associations entre ces entités, proposer un modèle EA sous forme graphique.

B. Requêtes SQL

Pour chacune des questions suivantes, répondre par une **requête SQL** donnant :

3. les noms des joueurs ayant participé au tournoi *Open de Paris* ;
4. pour chaque rencontre, les noms des deux joueurs et le nom du tournoi ;
5. le nombre de rencontres jouées dans chaque tournoi ;
6. les noms des joueurs ayant participé à au moins 3 tournois ;
7. le tournoi avec le plus grand nombre de rencontres ;
8. les joueurs ayant participé à un tournoi à Paris et à un tournoi à Lyon ;
9. les joueurs français ou ayant participé à un tournoi à Paris ;
10. les joueurs ayant participé à tous les tournois s'étant déroulés à Paris.

Mots-clés

Requête SQL ; sélection ; agrégation ; modèle entité–association ; jointures ; sous-requêtes ; opérations ensemblistes.

Plateforme de commerce en ligne

Thème abordé : bases de données.
Temps de résolution suggéré : 1 heure 15.

Présentation du devoir

On s'intéresse à la gestion d'une plateforme de commerce en ligne.

Des **clients** passent des **commandes**. Chaque commande est constituée de « **lignes** » qui correspondent à un **produit**, éventuellement acheté en plusieurs exemplaires. Chaque produit appartient à une **catégorie**. Les clients peuvent laisser des **avis** sur les produits.

Le schéma relationnel de la base de données de cette plateforme est le suivant :

```
CLIENT(idC, nom, ville)
COMMANDE(idCmd, dateCmd, idC)
PRODUIT(idP, nomP, prix, idCat)
CATEGORIE(idCat, nomCat)
LIGNE_COMMANDE(idCmd, idP, quantite)
AVIS(idC, idP, note)
```

Dans ce schéma, on repère des clés étrangères qui font référence ($\rightarrow$) à des clés primaires :

- `COMMANDE.idC` $\rightarrow$ `CLIENT.idC` ;
- `PRODUIT.idCat` $\rightarrow$ `CATEGORIE.idCat` ;
- `LIGNE_COMMANDE.idCmd` $\rightarrow$ `COMMANDE.idCmd` ;

— `LIGNE_COMMANDE.idP` → `PRODUIT.idP` ;

— `AVIS.idC` → `CLIENT.idC` ;

— `AVIS.idP` → `PRODUIT.idP`.

Questions

1. Proposer un diagramme Entité-Association (sans propriétés) modélisant la situation. Distinguer clairement les types d'entité et les types d'association.
2. Justifier la présence de la relation `LIGNE_COMMANDE` dans le schéma relationnel.

Pour les questions suivantes, écrire une **requête SQL** renvoyant :

3. pour chaque commande, le nom du client et la date de commande ;
4. les noms des clients ayant commandé un produit nommé *Ordinateur* ;
5. le montant total de chaque commande ;
6. le nombre total de produits distincts vendus par catégorie ;
7. les trois produits les plus chers (nom et prix) ;
8. le nom du client ayant dépensé le plus sur la plateforme et le montant total qu'il a dépensé ;
9. le nom et la note moyenne du produit ayant la meilleure note moyenne ;
10. les noms des clients ayant passé au moins une commande ou ayant rédigé au moins un avis ;
11. les noms des clients ayant dépensé plus que la dépense moyenne de tous les clients.

Mots-clés

Requête SQL ; sélection ; agrégation ; modèle entité–association ; jointures ; sous-requêtes ; opérations ensemblistes.

Plateforme d'écoute musicale

Thème abordé : bases de données.
Temps de résolution suggéré : 1 heure.

Présentation du devoir

On considère une plateforme de *streaming* musical.

Des **utilisateurs** écoutent des **morceaux** qui sont regroupés au sein d'**albums**. Chaque album est produit par un **artiste**. Les utilisateurs peuvent créer des **playlists** contenant plusieurs morceaux.

Le schéma relationnel de notre application est le suivant :

```
UTILISATEUR(idU, nom, pays)
ARTISTE(idA, nomA)
ALBUM(idAlb, titreAlb, idA)
MORCEAU(idM, titreM, duree, idAlb)
ECOUTE(idU, idM, dateEcoute)
PLAYLIST(idP, nomP, idU)
CONTIENT(idP, idM)
```

Dans ce schéma, on repère des clés étrangères qui font référence ($\rightarrow$) à des clés primaires :

- `ALBUM.idA` $\rightarrow$ `ARTISTE.idA`
- `MORCEAU.idAlb` $\rightarrow$ `ALBUM.idAlb`
- `ECOUTE.idU` $\rightarrow$ `UTILISATEUR.idU`

— `ECOUTE.idM` → `MORCEAU.idM`

— `PLAYLIST.idU` → `UTILISATEUR.idU`

— `CONTIENT.idP` → `PLAYLIST.idP`

— `CONTIENT.idM` → `MORCEAU.idM`

Questions

Pour toutes les questions, on demande d'écrire une **requête SQL**.

1. On veut les titres des morceaux de l'album *Greatest Hits*.
2. Pour chaque morceau écouté, on veut le nom de l'utilisateur et le titre du morceau.
3. On veut les noms des utilisateurs ayant écouté un morceau de l'artiste *Requête Man*, sans doublons.
4. On veut le nombre total d'écoutes pour chaque morceau.
5. On veut l'album contenant le plus grand nombre de morceaux.
6. On veut le titre et le nombre d'écoutes des 5 morceaux les plus écoutés.
7. On veut les noms des utilisateurs ayant effectué au moins 20 écoutes de morceaux sur la plateforme, ainsi que leur nombre d'écoutes.
8. On veut les noms des utilisateurs ayant créé au moins une playlist ou ayant écouté au moins un morceau.
9. On veut les noms des utilisateurs ayant créé au moins une playlist et ayant écouté au moins un morceau. Répondre de deux façons différentes (deux requêtes attendues séparément).

Mots-clés

Requête SQL ; sélection ; agrégation ; jointures ; opérations ensemblistes.

Gestion d'un festival de cinéma

Thème abordé : bases de données.
Temps de résolution suggéré : 1 heure.

Présentation du devoir

On souhaite gérer les données d'un festival de cinéma.

Description de la situation

— Des **réalisateurs** réalisent des **films**.
— Les films sont projetés lors de différentes **séances**.
— Des **spectateurs** assistent aux séances.
— Les spectateurs peuvent attribuer une **note** à un film.

Schéma relationnel

On donne ci-après le schéma relationnel de la base de données du festival.

```
REALISATEUR(idR, nomR, pays)
FILM(idF, titre, annee, idR)
SEANCE(idS, dateS, salle, idF)
SPECTATEUR(idSp, nomSp, ville)
ASSISTE(idSp, idS)
NOTE(idSp, idF, note)
```

Ce schéma n'est pas complet, dans la mesure où les clés primaires et étrangères ne sont pas mises en évidence. Cependant, les noms des attributs sont suffisamment explicites pour qu'il n'y ait pas d'ambiguïté.

Questions

On attend une **requête SQL** permettant de répondre à la problématique pour chacune des questions.

1. On souhaite le titre des films réalisés par un réalisateur nommé *Stanley Q. Lang*.
2. On souhaite afficher le nom des spectateurs et les titres des films auxquels ils ont assisté.
3. On souhaite les noms des spectateurs ayant assisté à une séance du film *Le Sage qui lisait*. Un spectateur qui aurait assisté à plusieurs séances de ce film ne doit pas apparaître plusieurs fois.
4. Pour chaque film, obtenir le nombre total de séances organisées.
5. Quel est le titre du film ayant obtenu le plus grand nombre de notes, et le nombre de notes associé ?
6. Quels sont les trois films ayant la meilleure note moyenne ?
7. Quels sont les films (avec leur moyenne) dont la note moyenne est strictement supérieure à la note moyenne globale de tous les films ?
8. On cherche les films de *Stanley Q. Lang* dont la note moyenne est strictement supérieure à la note moyenne globale de tous les films.
9. Enfin, on cherche les films de *Stanley Q. Lang* dont la note moyenne est strictement supérieure à la moyenne de tous ses films.
10. On cherche les réalisateurs dont au moins deux films différents ont été notés et dont la note moyenne globale des films est strictement supérieure à 7. On souhaite afficher le nom du réalisateur et sa moyenne globale, triés par moyenne décroissante.
11. Quels sont les noms des spectateurs ayant assisté à au moins une séance, mais n'ayant attribué aucune note ?

Mots-clés

Requête SQL ; sélection ; agrégation ; jointures ; sous-requêtes ; opérations ensemblistes.