Thèmes d'étude

1 Bases de données

Exercice 1 – (SQL-000)

Les classes de MPSI du lycée LMM organisent un tournoi d'intelligences artificielles de morpion. Une partie de morpion se déroule sur un damier de trois cases par trois cases. Un premier joueur choisit une case. Puis, le deuxième joueur choisit une autre case. Puis, le premier joueur choisit une troisième case, et l'on continue ainsi jusqu'à ce que :

- soit un joueur aligne trois cases, dans ce cas il gagne ;
- soit le damier est rempli, dans ce cas il y a match nul.

Les professeurs de ces classes ont donc créé une base de données afin de gérer ce tournoi. Cette base contient deux tables :

- une table contenant les informations des joueurs : identifiant du joueur, pseudonyme, date de naissance, classe ;
- une table contenant les informations des parties : identifiant de la partie, identifiant du joueur n° 1, identifiant du joueur n° 2, résultat de la partie, coups joués, date de la partie.

Voici les instructions SQL ayant permis de créé ces deux tables.

```
CREATE TABLE joueurs (
         idj INTEGER,
         pseudo VARCHAR(50),
         datenaiss DATE,
         classe VARCHAR(50),
         PRIMARY KEY (idj)
         );

CREATE TABLE parties (
         idp INTEGER,
         idj1 INTEGER,
         idj2 INTEGER,
         res INTEGER,
         coups INTEGER,
         date DATETIME,
         PRIMARY KEY(idp),
         FOREIGN KEY(idj1) REFERENCES joueurs,
         FOREIGN KEY(idj2) REFERENCES joueurs
         );
```

Quelques conventions et rappels.

- Si le résultat d'une partie vaut 0, il y a match nul, s'il vaut 1 le joueur 1 a gagné et s'il vaut 2 le joueur 2 a gagné.
- Pour une partie, on lit dans l'entier `coups` les coups effectués par les joueurs. Ainsi, si `coups = 54892`, alors
  - le joueur 1 joue dans la case 5 ;
  - le joueur 2 joue dans la case 4 ;
  - le joueur 1 joue dans la case 8 ;
  - le joueur 2 joue dans la case 9 ;
  - le joueur 1 joue dans la case 2.

  Ainsi, dans cette partie, le joueur 1 a gagné (alignement sur la deuxième colonne).
- Les dates (type `DATE`) sont au format `AAAA-MM-JJ`.
- Les dates-heures (type `DATETIME`) sont au format `AAAA-MM-JJ HH:MM:SS`.
- Les dates et dates-heures sont écrites comme des chaînes de caractères (exemple : `"1515-09-13"`) et peuvent être comparées entre elles.
- Si une requête ne donne pas qu'une réponse (exemple : plusieurs lignes vérifient un critère), on inscrira toutes les réponses.
- Pour effectuer une jointure d'une table `T` avec elle même, on pourra lui donner un alias avec l'instruction `AS`. Voici un exemple.

```
SELECT *
FROM T
JOIN T AS Tbis ON T.[...] = Tbis.[...] ;
```

Question 1
Quel est le pseudonyme de l'étudiant d'identifiant n° 42 ?

Question 2
Quel est l'identifiant de l'étudiant dont le pseudonyme est `"Galois"` ?

Question 3
Combien de parties ont été jouées ?

Question 4
Combien y a-t-il eu de matchs nuls ?

Question 5
Combien de parties différentes (*i.e.* de successions de coups différentes) ont été jouées ?

Question 6
Quel est l'identifiant de l'étudiant le plus jeune ? Les donner tous s'il y en a plusieurs.

Question 7
Combien de parties ont été jouées par l'étudiant de pseudonyme `"Tux"` ?

Question 8
Combien de parties ont été perdues par l'étudiant de pseudonyme `"Tux"` ?

Question 9
Combien de parties ont été gagnées par des étudiants de MPS1 contre des étudiants de MPS2 ?

Question 10
Quels sont les pseudonymes des joueurs ayant joué le plus de parties en tant que joueur n° 1 ? Les donner tous s'il y en a plusieurs.

Question 11
Combien de parties ont été jouées en sept coups ou moins, en 2018 et entre deux étudiants d'une même classe ?

Exercice 2 – (SQL-001)

La base de données[1] `medocs.sqlite` contient 5 tables :

LABORATOIRES : Contient une liste de laboratoires pharmaceutiques. Cette table possède deux attributs :
- **id :** Identifiant du laboratoire dans la base de données.
- **laboratoire :** Nom du laboratoire.

1. Source : http ://base-donnees-publique.medicaments.gouv.fr/telechargement.php

CIS_COMPO : Contient la liste des compositions qualitatives et quantitatives des médicaments de la base de données, substance par substance. Cette table contient 8 attributs :
- **code_CIS :** Code CIS (code identifiant de spécialité) d'un médicament, vous pouvez le considérer comme un identifiant de médicament.
- **désignation :** Désignation de l'élément pharmaceutique.
- **code_substance :** Code de la substance.
- **dénomination_substance :** Dénomination de la substance.
- **dosage :** Dosage de la substance.
- **ref_dosage :** Référence de ce dosage. Exemple : `"(pour) un comprimé"`.
- **nature_compo :** Nature du composant (principe actif : « SA » ou fraction thérapeutique : « ST »).
- **numéro_liaison :** Numéro permettant de lier, le cas échéant, substances actives et fractions thérapeutiques.

HAS_Liens : Contient les liens vers les avis de la commission de transparence (CT) de la Haute Autorité de la Santé (HAS). Cette table contient 2 attributs :
- **code_HAS :** Code de dossier HAS.
- **lien :** Lien vers la page d'avis de la CT.

CIS_bdpm : Cette table contient la liste des médicaments commercialisés, ou en arrêt de commercialisation depuis moins de trois ans. Cette table contient 11 attibuts :
- **code_CIS :** Code CIS (code identifiant de spécialité) d'un médicament, vous pouvez le considérer comme un identifiant de médicament.
- **dénomination :** Dénomination du médicament.
- **forme :** Forme pharmaceutique.
- **voie :** Voies d'administration (avec un séparateur «; » entre chaque valeur quand il y en a plusieurs).
- **statut :** Statut administratif de l'autorisation de mise sur le marché (AMM).
- **procédure :** Type de procédure d'autorisation de mise sur le marché (AMM).
- **commercialisation :** État de commercialisation.
- **date_AMM :** Date d'AMM (format JJ/MM/AAAA).
- **statutBdM :** Valeurs possibles : « Alerte » (icône rouge) ou « Warning disponibilité » (icône grise).
- **numéro_autorisation :** Numéro de l'autorisation européenne.
- **titulaire :** Numéro du laboratoire titulaire.
- **surveillance :** Surveillance renforcée (triangle noir) : valeurs « Oui » ou « Non ».

CIS_HAS_SMR : Cette table contient l'ensemble des avis de SMR (Service médical rendu) de la HAS. Cette table contient 6 attributs :
- **code_CIS :** Code CIS (code identifiant de spécialité) d'un médicament, vous pouvez le considérer comme un identifiant de médicament.
- **code_HAS :** Code de dossier HAS.
- **motif :** Motif d'évaluation.
- **date_avis :** Date de l'avis de la Commission de la transparence (format AAAAMMJJ).
- **valeur :** Valeur du SMR.
- **libellé :** Libellé du SMR.

Question.

Question 1
`code_CIS` est-il une clé primaire de la table `CIS_bdmp` ?

Question 2
`code_HAS` est-il une clé primaire de la table `CIS_HAS_SMR` ?

Question 3
Donner le nom du laboratoire dont le numéro d'identification est α.

Question 4
Donner le nombre de médicaments produits par ce laboratoire.

Question 5
Donner le nombre médicaments de ce laboratoire dont l'AMM a été donnée le 1er janvier 2000 ou après.

Question 6
Donner le code CIS du médicament de ce laboratoire ayant la plus ancienne AMM. S'il y en a plusieurs, on donnera le code CIS le plus petit.

Question 7
Quel est le lien internet vers la page d'avis de la CT sur ce dernier médicament (on ne donnera que la série de chiffres à la fin de cette adresse, qui sont au nombre de 6 ou 7 suivant les adresses) ?

Question 8
Quelle est la somme des numéros de liaison des médicaments de ce laboratoire ?

Question 9
Donner le code CIS du médicament de ce laboratoire ayant le plus de substances différentes. S'il y en a plusieurs, on donnera le code CIS le plus petit.

Question 10
Cette question peut se traiter en interrogeant la base de données depuis `Python` : combien de codes CIS contiennent votre numéro α comme sous-chaîne ? Si votre α est compris entre 0 et 9, vous le ferez précéder d'un 0. Par exemple les α valant 8 ou 27 sont contenus dans 60008927 mais pas dans 60002875.

Exercice 3 – (SQL-002)

Un professeur d'informatique a créé une base de données pour gérer les notes de ses interrogations hebdomadaires. Pour cela, il a créé trois tables : `etudiants`, `interros`, `notes`. Voici les commandes SQL ayant permis de créer ces tables.

```
CREATE TABLE etudiants (
    -- table des données des étudiants
    id INTEGER, -- identifiant de l'étudiant
␣␣␣␣nom␣VARCHAR␣NOT␣NULL,␣--␣nom␣de␣l'étudiant
    prenom VARCHAR NOT NULL, -- prénom de l'é
        tudiant
␣␣␣␣date_naissance␣DATE␣NOT␣NULL,␣--␣date␣de␣
    naissance␣de␣l'étudiant, format AAAA-MM-JJ
```

```
    PRIMARY KEY (id)
    );

CREATE TABLE interros (
    -- table des données des interros
    id INTEGER, -- identifiant de l'interro
␣␣␣␣titre␣VARCHAR,␣--␣titre␣de␣l'interro
    sujet VARCHAR, -- sujet de l'interro
␣␣␣␣date␣DATE,␣--␣date␣du␣jour␣où␣a␣été␣donnée␣
    l'interro, format AAAA-MM-JJ
    PRIMARY KEY (id)
    );

CREATE TABLE notes (
    -- table des notes des étudiants aux
        interros
    id_etudiant INTEGER NOT NULL, -- identifiant
         de l'étudiant
␣␣␣␣id_interro␣INTEGER␣NOT␣NULL,␣--␣identifiant
    ␣de␣l'interro
    note INTEGER NOT NULL, -- note obtenue
    PRIMARY KEY (id_etudiant, id_interro),
    FOREIGN KEY (id_etudiant) REFERENCES
        etudiants,
    FOREIGN KEY (id_interro) REFERENCES interros
    );
```

Question 1
Peut-on donner plusieurs notes au même étudiant et pour la même interrogation ?

Question 2
Donner une requête SQL traduisant l'opération suivante, exprimée dans le vocabulaire d'algèbre relationnelle usuel :

$$\pi_{\text{sujet}}(\sigma_{\text{id}=1}(\text{interros})).$$

Question 3
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants de la classe.

Question 4
Écrire une requête SQL permettant d'obtenir la liste des prénoms des étudiants de la classe, sans doublon.

Question 5
Les enfants du professeur se sont amusés à rentrer des données factices, que le professeur aimerait retrouver afin de les effacer ensuite. Écrire une requête SQL permettant d'obtenir la liste des identifiants des étudiants ayant pour nom `"reinedesneiges"`.

Question 6
Écrire une requête SQL permettant d'obtenir la date de naissance de l'étudiant le plus jeune de la classe.

Question 7
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants ayant obtenu au moins un 20 à une des interrogations.

Question 8
Écrire une requête SQL permettant d'obtenir la liste des noms, prénoms d'étudiants et titres d'interrogations pour chaque note de 0 obtenue.

Question 9
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants, avec la moyenne des notes de chaque étudiant.

Question 10
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants, suivis pour chaque étudiant du nombre d'interrogations rendues.

Question 11
Écrire une requête SQL permettant d'obtenir le nom, le prénom et le nombre de copies rendues par l'étudiant ayant rendu le plus de copies (on suppose qu'il n'y en a qu'un). On rappelle que l'instruction `LIMIT k` permet de tronquer une table à ses `k` premières lignes.

Question 12
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants ayant rendu au moins 10 copies, suivis du nombre de copies rendues.

Question 13
Écrire une requête SQL permettant d'obtenir la liste des titres des interrogations, suivie pour chaque interrogation du nombre d'étudiants qui n'ont pas rendu de copies.

Question 14
Écrire une requête SQL permettant d'obtenir la liste des titres des interrogations pour lesquelles tous les étudiants ont rendu une copie.

Exercice 4 – (SQL-003)

On s'intéresse dans cet exercice à la gestion des données produites par la caisse enregistreuse d'une boulangerie (fictive), un jour donné. Sur chaque ticket produit par la caisse figure une ligne par produit vendu, indiquant notamment le nombre d'unités vendues par produit et le prix de chaque produit.

Chaque étudiant dispose d'un fichier `bdd_boulangerie_`α`.sqlite` possédant trois tables. Voici les commandes ayant permis de créer ces tables.

```
CREATE TABLE tickets (
        -- table des tickets
        id INTEGER,
        heure TIME NOT NULL,
        paiement VARCHAR(10) NOT NULL,
        PRIMARY KEY (id)
        );

CREATE TABLE produits (
        -- table des produits
        id INTEGER,
        nom VARCHAR(50) NOT NULL,
        prix FLOAT,
        PRIMARY KEY (id)
        );

CREATE TABLE lignes_tickets (
        -- table des lignes des tickets
        id INTEGER,
        idt INTEGER,
        idp INTEGER,
        quantite INTEGER NOT NULL,
        PRIMARY KEY (id),
```

```
FOREIGN KEY (idt) REFERENCES tickets,
FOREIGN KEY (idp) REFERENCES produits
);
```

La table `produits` recence les produits vendus par la boulangerie et indique pour chaque produit son nom et son prix (en €).

La table `tickets` recence pour chaque ticket l'heure d'enregistrement du ticket et le moyen de paiment (carte bleue, liquide ou chèque).

La table `lignes_tickets` indique pour chaque ticket (identifiant `idt`) et chaque produit (identifiant `idp`) le nombre d'unités du produit acheté sur ce ticket.

Question 1
Combien d'unités ont été vendues par la boulangerie ce jour là ?

Question 2
Quel est le chiffre d'affaire de la boulangerie ce jour là ?

Question 3
Quel est l'identifiant du produit dont ont été vendues le plus d'unités ? S'il y en a plusieurs, mettez le plus petit.

Question 4
Combien de tickets ne contiennent qu'un produit vendu (quelqu'en soit le nombre d'unités) ?

Question 5
Combien de tickets ne contiennent que des produits vendus en une unité ?

Question 6
Quel est l'heure du dernier ticket enregistré ?

Question 7
Quelle est la valeur en € du premier ticket enregistré ?

La banque de la boulangerie prélève une commission de 1% sur chaque transaction effectuée par carte bleue.

Question 8
Quel est le montant total prélevé par la banque à la boulangerie ce jour là ?

Question 9
Combien y a-t-il de tickets dont la valeur est supérieure ou égale à 10 € ?

Exercice 5 – (SQL-004)

Donner des requêtes pour :

- trouver le nombre d'acteurs qui ont joué au moins deux rôles ;
- trouver les acteurs qui sont aussi des réalisateurs ;
- trouver les acteurs nés le 1er janvier 1930 ou après ;
- compter les acteurs nés avant le 1er janvier 1940 ;
- trouver les acteurs qui ont joué au moins dans un film où jouait Martin Freeman ;
- trouver les réalisateurs qui ont dirigé Clint Eastwood.

Exercice 6 – (SQL-005)

Base de données des Pokemon

Nous allons utiliser la base de données issue du site `http://veekun.com/`. Un fichier nommé « veekun-pokedex.sqlite » doit être présent sur le bureau de votre ordinateur. Ouvrir cette base de données avec DB Browser for SQLite.

Structure de la table de données

Question 1
En utilisant DB Browser for SQLite, donner le nombre de tables contenu dans la base de données.

Dans un premier temps, nous allons utiliser uniquement la table `pokemon` qui répertorie les pokémons.

Question 2
Donner le schéma relationnel de cette table. On le donnera sous la forme nom_table(attribut_1 : type, attribut_2 : type, ...).

Question 3
Donner la définition d'une clé primaire.

Table des pokemons

Question 4
Quelle est la taille de pikachu ?

Question 5
Quelle est le poids de pikachu ?

Question 6
En utilisant une des valeurs précédentes, quels pokemons sont plus grands (strictement) que pikachu ?

Question 7
Sans utiliser une des valeurs précédentes, quels pokemons sont plus grands (strictement) que pikachu ?

Question 8
Combien y a t-il de pokemons plus grands (strictement) que pikachu ?

Question 9
Combien de pokemons ont la même taille que pikachu (lui y compris) ?

Question 10
Parmi les pokemons ayant la même taille que pikachu, donner le nom et le poids du plus gros.

Question 11
Donner le nom et la taille et le poids du plus grand pokemon.

Question 12
Quel pokémon est le plus petit ? (Il peut y en avoir plusieurs ...)

Question 13
Lister le nombre de pokemons par taille en les classant du plus grand au plus petit.

Question 14
Quel est le nombre maximal de pokemons ayant la même taille ? Donner la taille et le nombre.

Question 15
Quels est le nom et la taille du second pokemon le plus grand ?

Question 16
Quelle est la taille moyenne des pokemons ? (deux décimales après la virgule).

Notre niveau d'expérience permet d'attraper des pokemons de hauteur égale à 8 à 0.5 près (inclus). Nous souhai-

terions donc savoir combien de pokemons pourront être attrapés sans changer ces réglages.

Question 17
Combien de pokémons sont capturés avec le réglage par défaut ?

Classement des pokemons

Maintenant nous souhaitons placer les pokemons sur une carte selon leurs propriétés, pour cela nous allons utiliser les tables suivantes.

La table `pokemon_species` contient les colonnes :

- `id` (clé primaire) : identifiant du pokemon ;
- `identifier` : nom du pokemon ;
- `generation_id` : identifiant de génération qui correspond aussi au numéro de la génération ;
- `habitat_id` : identifiant d'habitat ;

D'autres attributs existent, mais ils ne seront pas utilisés dans notre étude.

La table `pokemon_habitats` contient les colonnes :

- `id` (clé primaire) : identifiant d'habitat ;
- `identifier` : nom de l'habitat ;

Question 18
Écrire la requête SQL permettant d'afficher le nom du pokemon et le nom de son habitat.

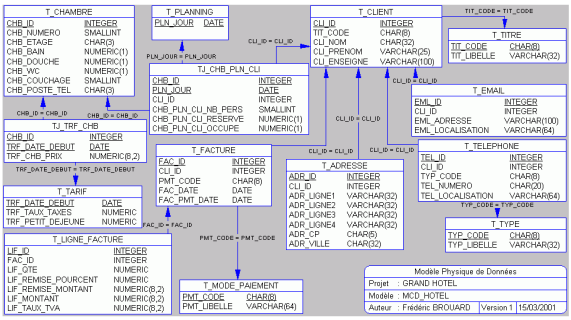
Question 19
Combien de pokemons vivent en forêt ('forest' en anglais) ?

Question 20
Combien de pokemons de la generation 3 vivent en forêt ('forest' en anglais) ?

Exercice 7 – (hotel)

On donne le schéma d'une base de données d'un hôtel.

Le modèle physique de la base de données d'un hotel est le suivant.



La plupart des clefs sont des entiers (I) qui pourront être auto générés par exemple par auto-incrément. Pour certaines entités, notamment celles servant de références à la saisie (`MODE_PAIEMENT`, `TYPE`) la clef est un code. Enfin pour les entités `TARIF` et `PLANNING`, il a été choisi une date comme clef. Chaque entité est repérée à l'aide d'un trigramme (code de 3 lettres) qui sert de préfixe pour chaque attribut. Exemple : `CHB` pour `CHAMBRE`, `LIF` pour `LIGNE_FACTURE`, etc... Les booléens seront représentés par des valeurs numériques 0 (faux) et 1 (vrai), chaque attribut ayant obligatoirement une valeur par défaut. L'association « occupée » permet de connaître la réservation ou l'occupation d'une chambre (une chambre peut avoir été réservée mais pas occupée), c'est pourquoi cette association possède les attributs `NB_PERS` (nombre de personnes : entier) `RESERVE` (réservée : booléen) et `OCCUPE` (occupe : booléen).

Une chambre à une date donnée, ne peut être occupée que par un seul client. Mais un client peut occuper plusieurs chambres à la même date ou la même chambre à différentes dates, voire même plusieurs chambres à plusieurs dates...

Définition des entités :

- l'entité `CLIENT` : un client peut avoir plusieurs adresses, plusieurs numéros de téléphone et plusieurs e-mail. Pour le téléphone, comme pour l'e-mail, l'attribut `localisation` permet de savoir si le téléphone est situé au domicile, à l'entreprise, etc...
- l'entité `TITRE` permet de donner un titre à une personne, parmi les valeurs `M.` (monsieur), `Mme.` (madame) et `Melle.` (mademoiselle) ;
- l'entité `TYPE` permet de connaître le type de téléphone, parmi les valeurs `TEL` (téléphone), `FAX` (télécopie) et `GSM` (portable) ;
- l'entité `MODE_PAIEMENT` permet de connaître le genre de paiement, parmi les valeurs `ESP` (espèces), `CHQ` (chèque), `CB` (carte bancaire). L'association « payée » intègre la date du paiement d'une facture.

Travail demandé

Question 1 *Donner la requête permettant de lister tous les noms, prénoms et les titres (M. Mme. ou Melle.) des clients.*

Question 2 *Donner le nombre de clients enregistrés dans la base.*

Question 3 *Trouver les noms et prénoms des clients dont le titre est 'Mme.' (madame).*

Question 4 *Donner les noms , prénoms et les titres (Mme. ou Melle.) des clientes.*

Question 5 *Donner le nombre de clientes.*

Question 6 *Classer par ordre alphabétique les clients de sexe féminin. On ne demande que les noms et les prénoms qui devront être appelés Noms et Prénoms.*

Question 7 *Établir une nouvelle relation (table) faisant apparaître les noms des clients et leurs numéros de téléphone.*

Question 8 *Donner la liste des clients ayant le même nom*

Question 9 *Donner le nombre d'occurrences de chacun de ces noms.*

Question 10 *Donner la valeur moyenne des remises en pourcentage et en montant.*

Question 11 *Donner la valeur maximale des remises*

en pourcentage et en montant.

Question 12 *Donner les identifiants facture (*`FAC_ID` *que l'on renomera* `fac_id1`*) qui ont bénéficié de remise (soit en pourcentage soit en montant).*

Question 13 *Donner les identifiants clients (*`CLI_ID`*) qui ont bénéficié d'une remise (soit en pourcentage soit en montant).*

Question 14 *Donner les identifiants clients (*`CLI_ID`*) qui n'ont pas bénéficié d'une remise.*

Question 15 *Donner le nom et prénom du client qui a le plus bénéficié de remises. Donner le montant de cette remise totale.*