

Evaluation finale

Durée : 3 heures

Barème : 20 points

Partie 1 : Python et Peewee (10 points)

Exercice 1.1 : Modèles Peewee (3 points)

Créez un fichier `evaluation.py` qui définit les **modèles Peewee** correspondant à la base de données PetCare.

Vous devez définir les modèles suivants :

- `Proprietaire`
- `Veterinaire`
- `Animal`
- `Consultation`
- `Traitement`
- `ConsultationTraitement`

Chaque modèle doit :

- Hériter d'un `BaseModel` connecté à la base MySQL `petcare`
- Avoir les bons types de champs (`CharField`, `IntegerField`, `DecimalField`, `DateField`, `TextField`...)
- Déclarer les **clés étrangères** avec `ForeignKeyField`
- Avoir un `table_name` correspondant au nom de la table SQL

Exercice 1.2 : CRUD avec Peewee (4 points)

Dans le même fichier, écrivez le code pour réaliser les opérations suivantes :

Create (1 point)

1. Ajoutez un nouveau propriétaire : **Lemoine Camille**, téléphone `0656781234`, email `camille.lemoine@mail.com`, ville `Bordeaux`.
2. Ajoutez-lui un animal : **Pixel**, un chat de race **Bengal**, né le **2023-06-15**, pesant **4.5 kg**.
3. Créez une consultation pour Pixel, le **2025-05-15**, motif "Première visite", réalisée par le **Dr. Girard** (id=3).
4. Associez les traitements **Vermifuge** (id=3) et **Puce électronique** (id=10) à cette consultation.

Enveloppez le tout dans une **transaction** (`db.atomic()`).

Read (1 point)

5. Affichez tous les animaux du propriétaire **Martin Claire**, avec leur espèce et leur poids.
6. Affichez toutes les consultations de **Rex** (animal id=1) avec la date, le motif et le nom du vétérinaire.

7. Pour chaque espèce, affichez le nombre d'animaux et le poids moyen (utilisez `fn.COUNT` et `fn.AVG`).

Update (1 point)

8. Le poids de **Buddy** (animal id=3) a changé. Mettez-le à jour à **29.50 kg**.

Delete (1 point)

9. Supprimez le poisson **Nemo** (animal id=6). Attention : il faut d'abord vérifier qu'il n'a pas de consultations associées. Si c'est le cas, affichez un message d'avertissement. Sinon, supprimez-le.

Exercice 1.3 : Requêtes avancées avec Peewee (3 points)

10. **(1 point)** Affichez le **top 3 des vétérinaires** par nombre de consultations, en utilisant une jointure et une agrégation. Affichez le nom du vétérinaire, sa spécialité, et le nombre de consultations.
11. **(1 point)** Affichez la **facture détaillée** de la consultation n°1 :
- Date de la consultation
 - Nom de l'animal et nom du propriétaire
 - Nom du vétérinaire
 - Prix de la consultation
 - Liste des traitements avec dosage et prix
 - **Total** (consultation + traitements)
12. **(1 point)** Écrivez une **fonction** `historique_animal(animale_id)` qui prend un identifiant d'animal et affiche :
- Les informations de l'animal (nom, espèce, race, poids)
 - Le nom de son propriétaire
 - La liste de toutes ses consultations (date, vétérinaire, motif, diagnostic)
 - Pour chaque consultation, les traitements administrés (nom, dosage, prix)
 - Le **coût total** de toutes les consultations et traitements pour cet animal

Testez cette fonction avec l'animal id=1 (Rex).

Partie 2 : SQL — Base de données Datacenters (10 points)

Mise en place

Le formateur vous fournira le fichier `setup_datacenters.sql` qui contient le script de création et de peuplement de la base de données. Exécutez-le dans MySQL avant de commencer.

La base contient les tables suivantes :

- **datacenter** (id, nom, ville, pays, capacite_racks, annee_ouverture)
- **salle** (id, nom, etage, superficie_m2, nb_racks, datacenter_id)
- **client** (id, nom, email, telephone, type_contrat)
- **serveur** (id, hostname, marque, modele, cpu_cores, ram_go, stockage_to, date_installation, salle_id)
- **intervention** (id, date_intervention, type_intervention, description, duree_heures, cout, serveur_id)

- **hébergement** (id, date_debut, date_fin, prix_mensuel, serveur_id, client_id)

Exercice 2.1 : Requêtes de base (3 points)

1. Affichez le nom et la ville de tous les datacenters situés en **France**, triés par année d'ouverture.
2. Affichez les serveurs de marque **Dell**, avec leur hostname, modèle et nombre de cœurs CPU. Triez par hostname.
3. Affichez les interventions qui ont eu lieu en **mars 2025**, avec le type d'intervention et le coût.

Exercice 2.2 : Jointures (4 points)

1. Affichez la liste de tous les serveurs avec le nom de la salle et le nom du datacenter où ils se trouvent. Triez par nom de datacenter puis par hostname.
2. Affichez tous les hébergements en cours (ceux dont la **date_fin** est **NULL**) avec le nom du client, le hostname du serveur et le prix mensuel. Triez par prix mensuel décroissant.
3. Affichez les serveurs qui n'ont **jamais eu d'intervention**. Affichez leur hostname et leur date d'installation. (Indice : utilisez un LEFT JOIN)
4. Affichez les clients avec le nombre de serveurs qu'ils hébergent et le coût mensuel total de leurs hébergements. Triez par coût mensuel total décroissant.

Exercice 2.3 : Agrégations (3 points)

1. Comptez le **nombre de serveurs par marque**.
2. Calculez le **coût total des interventions** par type d'intervention.
3. Affichez les datacenters qui hébergent **plus de 3 serveurs**, avec le nombre de serveurs.
4. Affichez le mois (année-mois) où il y a eu le plus d'interventions, avec le nombre d'interventions et le coût total de ce mois.

Critères d'évaluation

| Partie | Points | Critères |
|------------------------|-----------|---|
| Python / Peewee | 10 | Code fonctionnel, bonne utilisation de l'ORM, gestion des erreurs |
| SQL Datacenters | 10 | Requêtes fonctionnelles, syntaxe correcte, résultats attendus |
| Total | 20 | |

Conseils

- Lisez **toutes** les questions avant de commencer pour avoir une vue d'ensemble
- Testez chaque requête SQL **individuellement** avant de passer à la suivante
- Pour la partie Peewee, vérifiez que votre connexion à la base fonctionne avant de coder les requêtes
- N'oubliez pas les **try/except** pour gérer les erreurs (DoesNotExist, IntegrityError...)
- Utilisez **db.atomic()** pour les transactions quand c'est demandé