ALIASE
IA - SERVICE & FORMATION

Formation Python

Atelier Pratique AP-PY 13

ALIASE

SQLite

Atelier Pratique AP-PY13 : Exercice PY-13.3

ALIASE
IA - SERVICE & FORMATION

SQLite : Charger un fichier excel dans une base de données

a) Définir une fonction **connect()** : connexion à la base **'effectif.db'** et retourner l'identifiant **(con = sqlite3.connect)**

b) Définir une fonction **lire_excel()** : Charger le fichier excel **'emp.xlsx'** en mémoire et retourner le tableau **tab_emps**

- **data = pd.read_excel("emp.xlsx")**
- **tab_emps = data.values (tableau des données sans les indexes)**

c) Définir une fonction **create_table() :** Créer une table **'emp'** avec les mêmes noms de colonnes dans fichier excel

d) Définir une fonction **load_table(tab_emps) : I**nsérer toutes les lignes du tableau **tab_emps** dans la table **'emp'**

e) définir une fonction **afficher_table()** qui affiche l'ensemble des données dans la table 'emp'

f) définir une fonction **afficher_group_by_sexe() :** Afficher le **nombre d'employés par sexe**

g) définir une fonction **afficher_group_by_dept() :** Afficher le **salaire moyen par département**

Atelier Pratique AP-PY13 : Exercice PY-13.3 (Cont)

ALIASE
IA - SERVICE & FORMATION

h) Définir une fonction **update_salaire() : ajouter 1000 € aux employés dont le salaire est < 2500**

i) **Programme principal** : Appeler les fonctions

```
con = connect()
cursor = con.cursor()

. . .

con.close()
```

Solution → Voir PYExo13.3.py

Atelier Pratique AP-PY13 : Exercice PY-13.6

ALIASE
IA - SERVICE & FORMATION

SQLite : Modifier le programme **PY-13.3** en intégrant le contrôle des erreurs

```
try:
        . . .
except (Exception, sqlite3.DatabaseError) as error:        # Ce bloc sera exécuté en cas d'erreur
        . . .
finally:                                                   # Ce bloc sera exécuté quoi qu'il arrive
        . . .
```

Solution → Voir PGExo13.6.py

Atelier Pratique AP-PY13 : Exercice PY-13.7

ALIASE
IA - SERVICE & FORMATION

SQLite : Modifier le programme **PY-13.3** comme ceci :

- Générer la requête **'CREATE TABLE …. '** avec les mêmes noms et types des colonnes dans le fichier Excel

- Fonction ***lire_excel(ficname)*** , sera appelée avec le paramètre **"emp.xlsx"** , et retournera le ***df*** (Dataframe)

- Fonction ***create_table(tabname, df)*** , sera appelée avec les 2 paramètres **"emp" et df**

Solution → Voir PGExo13.7.py

PostgreSQL

Atelier Pratique AP-PY13 : Exercice PG-13.3

ALIASE
IA - SERVICE & FORMATION

PostgreSQL : Charger un fichier excel dans une base de données

Refaire l'exercice **PY-13.3** (SQLite) mais en utilisant la base de données **PostgreSQL**

Solution → Voir PGExo13.3.py

Atelier Pratique AP-PY13 : Exercice PG-13.6

ALIASE
IA - SERVICE & FORMATION

PostgreSQL : Modifier le programme **PG-13.3** en intégrant le contrôle des erreurs

```
try:
        . . .
except (Exception, psycopg2.DatabaseError) as error:        # Ce bloc sera exécuté en cas d'erreur
        . . .
finally:                                                    # Ce bloc sera exécuté quoi qu'il arrive
        . . .
```

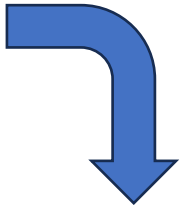
Solution → Voir PGExo13.6.py

Atelier Pratique AP-PY13 : Exercice PG-13.8

ALIASE
IA - SERVICE & FORMATION

Objectif : **Importer** deux tables ***tab1*** et ***tab2*** à partir des deux fichiers **csv**

Voir les **étapes** et **solution** dans le slide suivant



Atelier Pratique AP-PY13 : Exercice PG-13.8

ALIASE
IA - SERVICE & FORMATION

1) **Source PG-13.8.0-create_tabs.py :** Définir une fonction **create_table() : Créer deux tables *tab1* et *tab2***

2) **Source PG-13.8.1-Excel-CSV-copy_from.py**

Définir une fonction **create_csv()** :

- **Lire 'File3.xlsx', FEUILLE-1** dans un Dataframe df puis **créer le fichier *Feuille1.csv* sans l'entête**
- **Lire 'File3.xlsx', FEUILLE-2** dans un Dataframe df puis **créer le fichier *Feuille2.csv* sans l'entête**

Définir une fonction **load_tables()** :

- **Charger *tab1* à partir des données dans le fichier *Feuille1.csv* , en utilisant copy_from()**
- **Charger *tab2* à partir des données dans le fichier *Feuille2.csv* , en utilisant copy_from()**

Exécuter la séquence suivante :

- **Créer deux tables *tab1* et *tab2***
- **Créer les deux fichiers csv à partir des deux onglets Excel**
- **Importer les deux tables *tab1* et *tab2* à partir des deux fichiers csv**
- **Afficher le contenu des deux tables**

3) **Source PG-13.8.2-SELECT.py :** Définir une fonction **afficher_table() : Afficher le contenu des deux tables *tab1* et *tab2***

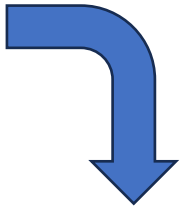
Atelier Pratique AP-PY13 : Exercice PG-13.9

ALIASE
IA - SERVICE & FORMATION

Objectif :

Mesurer et comparer les **performances** des commandes **COPY, copy_from, copy_expert**

Voir les **étapes** et **solution** dans le slide suivant



Atelier Pratique AP-PY13 : Exercice PG-13.9

ALIASE
IA - SERVICE & FORMATION

1) **Source** PG-13.9.0-create_data_file.py : Définir une fonction **create_file(NB_ROWS) :**

   - créer une ***Liste*** de tuples **[('prenom-1', 'nom-1', 1) . . . ('prenom-i', 'nom-i', 1)]**
   - créer le fichier ***bigdata.csv*** sans entête à partir de la ***Liste*** , et le placer dans le dossier ***'C:\Users\Public'***

2) **Source** PG-13.9.1-PERF-COPY-copy_from.py

   - définir une fonction **create_table()** : CREATE TABLE tab3 (prenom, nom, age)
   - définir une fonction **mesure_COPY()** : mesure le temps d'exécution du COPY
   - définir une fonction **mesure_copy_from()** : mesure le temps d'exécution du copy_from

3) **Source** PG-13.9.2-PERF-COPY-copy_expert.py

   - définir une fonction **mesure_COPY()** : mesure le temps d'exécution du COPY
   - définir une fonction **mesure_copy_expert()** : mesure le temps d'exécution du copy_expert

4) **Source** PG-13.9.3-PERF-copy_from-copy_expert.py

   - définir une fonction **mesure_copy_from()** : mesure le temps d'exécution du copy_from
   - définir une fonction **mesure_copy_expert()** : mesure le temps d'exécution du copy_expert