Atelier 02 : Processus ETL avec Python

1. Objectif et instructions:

L'objectif de cet atelier est de vous familiariser avec les concepts de l'ETL avec Python.

- Ce Tp est noté.
- Utilisez VS Code pour développer vos solutions.
- Chaque exercice doit être soumis sous la forme d'un notebook Jupyter nommé selon le format suivant : ‘atelier02_ex1.ipynb’. Adaptez le nom du fichier pour les autres exercices .

2. Cas d’utilisation:

ASC organise un événement annuel pendant les vacances d'hiver. Ce séjour d'une semaine à la montagne propose une multitude d'activités telles que le ski, le snowboard, des soirées festives, etc.

Lorsqu'un étudiant souhaite participer à cet événement, les organisateurs enregistrent ses informations personnelles, ainsi que la date et les frais d'inscription dans un fichier Excel. L'étudiant peut payer les frais d'inscription immédiatement ou plus tard. Une fois le paiement effectué, la date de paiement est ajoutée dans le fichier Excel.

Les organisateurs utilisent deux fichiers Excel : l'un contient les données d'inscription, y compris les informations personnelles des étudiants (matricule, nom, prénom, sexe, et adresse électronique), les dates d'inscription et de paiement, les frais d'inscription, et l'année de l'événement. L'autre fichier Excel répertorie les informations sur les membres de l'association ASC.

3. ETL & BD:

Nous voulons développer un logiciel, que les organisateurs peuvent utiliser pour gérer les inscriptions de manière efficace. Ce logiciel utilise une base de données relationnelle pour stocker les données. Deux modules sont utilisés pour créer la base de données et y importer ces données :

- Le module BD est utilisé pour créer la base de données et ses tables.
- Le module ETL est utilisé pour importer (dans la base de données) les données sur les éditions passées qui sont conservées dans les deux feuilles de calcul mentionnées ci-dessus.

Exercice 1 :

Nous voulons utiliser SQLite comme système de gestion de base de données relationnelle. Ce n'est pas le cas des SGBD plus sophistiqués, où une base de données est constituée de plusieurs fichiers, une base de données SQLite consiste en un seul fichier.

1. Le code suivant ouvre le fichier ‘./data/airports.db’. Ce fichier est une base de données SQLite qui contient une table nommée ‘airport’.

```
import sqlite3
conn = sqlite3.connect('./data/airports.db')
```

   Exécutez le code.

2. Pour exécuter des requêtes SQL sur la base de données, nous devons obtenir un 'curseur' de la connexion :

```
cur = conn.cursor()
```

   Exécutez le code.

3. Nous pouvons appeler la fonction 'execute()' sur le curseur 'cur' pour exécuter n'importe quelle instruction SQL.

```
res = cur.execute('''
                  SELECT airport_name
                  FROM airport
                  WHERE airport_id = 1382
                  ''')
```

a) Exécutez et testez le code. Donnez une interprétation.

b) Nous nous attendons à n'avoir qu'une seule ligne comme résultat. En utilisant la fonction ‘fetchone()‘ donnez l’instruction permettant d’afficher cette ligne.

4. Donnez les instructions pour décrire la table airport. Nous nous attendons le résultat suivant :

```
(0, 'airport_id', 'INTEGER', 0, None, 1)
(1, 'airport_name', 'TEXT', 0, None, 0)
(2, 'city', 'TEXT', 0, None, 0)
(3, 'country', 'TEXT', 0, None, 0)
(4, 'airport_iata', 'TEXT', 0, None, 0)
(5, 'airport_icao', 'TEXT', 0, None, 0)
(6, 'latitude', 'REAL', 0, None, 0)
(7, 'longitude', 'REAL', 0, None, 0)
```

5. Nous souhaiterions connaître le nom de tous les aéroports de Paris, France. Écrivez la requête SQL correspondante et envoyez-la au SGBD.
6. Écrivez la même requête que ci-dessus, mais au lieu de sélectionner uniquement le nom de l'aéroport, vous sélectionnez toutes les colonnes. Combien d'éléments contient chaque tuple dans le résultat ?
7. Dans un processus ETL, la sécurité des données est primordiale à chaque étape. L'utilisation de requêtes paramétrées pour éviter les injections SQL est particulièrement importante dans les phases d'extraction et de chargement des données.

   L'injection SQL est une technique d'attaque qui permet à un utilisateur malveillant d'exécuter des commandes SQL arbitraires sur une base de données en manipulant les entrées utilisateur. Cela peut entraîner des fuites

de données, des suppressions de données, ou d'autres actions non désirées sur la base de données. Exécutez et testez le code suivant :

```
country_name = input("Saisir le nom d'un pays (en anglais) : ")

sql_query = f"SELECT airport_name, country FROM airport WHERE country='{country_name}'"

print("\nRequête : ", sql_query)

print("\nRésultat de la requête : ")

res = cur.execute(sql_query)
for row in res:
    print(row[0], "---", row[1])
```

✓ 4.8s

```
Requête :  SELECT airport_name, country FROM airport WHERE country='France'

Résultat de la requête :
Calais-Dunkerque Airport --- France
Péronne-Saint-Quentin Airport --- France
Nangis-Les Loges Airport --- France
'"Bagnoles-de-l''Orne-Couterne Airport"' --- France
Albert-Bray Airport --- France
'"Le Touquet-Côte d''Opale Airport"' --- France
Valenciennes-Denain Airport --- France
Amiens-Glisy Airport --- France
```

8. Si un utilisateur malveillant saisit la valeur France'; DROP TABLE airport; --, le code SQL généré est :

```
country_name = input("Saisir le nom d'un pays (en anglais) : ")

sql_query = f"SELECT airport_name, country FROM airport WHERE country='{country_name}'"

print("\nRequête : ", sql_query)

print("\nRésultat de la requête : ")

#res = cur.execute(sql_query)
#for row in res:
#    print(row[0], "---", row[1])
```

✓ 3.6s

```
Requête :  SELECT airport_name, country FROM airport WHERE country='France'; DROP TABLE airport; --'
```

Cette requête va non seulement récupérer les aéroports de France, mais aussi supprimer la table airport de la base de données, ce qui est une action destructrice.Quelle leçon à retenir ?

9. La solution pour prévenir cette vulnérabilité est d'utiliser la substitution de paramètres (aussi appelée requêtes paramétrées ou préparées). Cela permet de séparer le code SQL de la donnée, empêchant ainsi toute injection SQL. Donnez les instructions de cette solution.
10. Fermer la connexion.
11. Créez une nouvelle base de données en passant à la fonction connect le fichier politics.db (qui n'existe pas encore dans ./data).
12. Ecrivez le code pour créer la table suivante : ’party(party_id INTEGER PRIMARY KEY, party_name TEXT NOT NULL, party_acronym TEXT, party_country TEXT NOT NULL)’

    Il ne peut y avoir deux partis avec le même nom dans le même pays. Imposez une contrainte UNIQUE sur ‘{party_name, party_country}’.

13. Écrivez le code pour insérer une ligne spécifiée (1, 'Les Républicains', 'LR', 'France') dans la table 'party’. N'oubliez pas d'utiliser la substitution de paramètres. N'oubliez pas d'utiliser la substitution de paramètres.
14. Vérifiez si la ligne a été correctement insérée dans la table à l'aide d'une instruction SELECT.
15. Une transactio est une séquence d'instructions de lecture et d'écriture sur la base de données qui sont considérées comme une opération atomique.

    En d'autres termes, l'effet de chaque instruction individuelle n'est visible dans la base de données que lorsque (et si) toutes les instructions sont exécutées avec succès. Si une seule instruction de la transaction échoue, l'effet de toutes les autres instructions est également annulé. Lorsqu'une opération INSERT est exécutée, SQLite démarre automatiquement une nouvelle transaction. Si nous voulons que le résultat de cette opération INSERT soit visible dans la base de données, nous devons explicitement valider la transaction avec ‘commit’.

    Exécutez *conn.commit()* et vérifiez que la nouvelle ligne apparaît maintenant dans la table 'party'.

Exercice 2 :

Pour cet exercice vous pouvez consulter la documentation :
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

1. Écrivez un script Python utilisant la bibliothèque pandas pour lire le fichier de données tabulées (gapminder.tsv) et afficher le contenu de ce fichier. Le fichier est situé dans le répertoire ./data. Utilisez la fonction read_csv de pandas avec le bon séparateur.

   Utilisez l'argument sep='\t' dans la fonction read_csv pour indiquer que les colonnes du fichier sont séparées par des tabulations.

2. Un DataFrame Pandas se compose de trois parties. L'index : les noms des lignes, les columns : les noms des colonnes et les values : les valeurs des données.

   Après avoir lu le fichier gapminder.tsv dans un DataFrame avec pandas, affichez les informations suivantes : Les index, les noms des colonnes. et les valeurs contenues dans le DataFrame.

   Écrivez le code Python nécessaire pour obtenir et afficher ces informations.

3. Affichez les cinq premières lignes du DataFrame.
4. Affichez les cinq dernières lignes du DataFrame.
5. Extrayez la colonne country du DataFrame et affichez-la et leur type.
6. Afficher la dernière ligne du DataFrame.
7. Sélectionnez et affichez les lignes correspondant aux index 0, 222, et 212.
8. Afficher la dernière ligne du DataFrame avec iloc.
9. Utilisez 'loc' pour sélectionner toutes les lignes et seulement les colonnes 'country' et 'continent'.
10. Faire exactement la même chose (question 9) avec iloc.
11. En utilisant les fonctions Python 'range' et 'iloc', pouvez-vous sélectionner les 3 premières colonnes du DataFrame.

12. Ayant travaillé avec SQL, vous êtes déjà conscient de l'importance de l'agrégation et du regroupement dans l'analyse des données. La méthode 'groupby()' permet de regrouper les lignes d'un DataFrame en fonction des valeurs de certaines colonnes. Testez et interprétez le code suivant :

grouped_year = df.groupby('year')

print(grouped_year)

13. Groupez les données par la colonne year et affichez la moyenne des colonnes numériques pour chaque année. Expliquez le rôle de l'argument numeric_only=True dans la méthode mean.
14. Groupez les données par la colonne year et affichez la moyenne de la colonne ‘lifeExp’ pour chaque année (l'espérance de vie moyenne).
15. Écrivez le code pour assigner à la variable ‘life_gdp’ un DataFrame qui contient l'espérance de vie moyenne et le PIB par habitant par année et par continent
16. Testez et interprétez le code suivant :

```
continent_col = df['continent']

print(f"Première valeur dans la colonne : {continent_col.loc[0]}")

print(f"Type des valeurs dans la colonne : {continent_col.dtype}")

print(f"Nombre de valeurs dans la colonne : {continent_col.size}")

print(f"Dimension d'une colonne : {continent_col.shape}")
```

17. Testez et interprétez le code suivant :

```
life_exp_col = df['lifeExp']

print(f"Espérance de vie maximale : {life_exp_col.max()}")
print(f"Espérance de vie minimale : {life_exp_col.min()}")
print(f"Espérance de vie moyenne : {life_exp_col.mean()}")
print(f"Écart-type de l'espérance de vie : {life_exp_col.std()}")
print(f"Médiane de l'espérance de vie : {life_exp_col.median()}")
```

18. Affichez toutes les statistiques importantes sur un 'Series' en une seule instruction.
19. Testez et interprétez les codes suivants :

```
life_exp_col < life_exp_col.mean()
```

```
print(life_exp_col[life_exp_col < life_exp_col.mean()])
```

```
print(life_exp_col + life_exp_col)
```

```
print(life_exp_col + 10)
```

20. Écrivez un code pour obtenir une 'Series' où les valeurs d'espérance de vie sont normalisées entre 0 et 1.
21. Écrivez un code pour assigner à la variable 'below_average_df' un DataFrame avec une colonne contenant les noms des pays qui ont ou ont eu par le passé une espérance de vie inférieure à la moyenne.

Exercice 3 :

1. Écrivez un script Python utilisant la bibliothèque pandas pour lire le fichier de données tabulées (books.tcsv) et afficher le contenu de ce fichier. Le fichier est situé dans le répertoire ./data. Utilisez la fonction read_csv de pandas avec le bon séparateur.
2. Quel est le contenu de la colonne ‘language_code’ dans le DataFrame ?
3. Listez les codes de langue qui contiennent "en" dans la colonne language_code, tout en éliminant les doublons ?
4. Écrivez un code pour obtenir un nouveau DataFrame, où toutes les valeurs qui font référence à la langue anglaise dans la colonne language_code sont remplacées par 'en'.
5. La colonne 'publication_date' contient des dates. Toutes les dates sont au format mois/jour/année. Cependant, le format de la date n'est pas cohérent d'une ligne à l'autre (le jour ou le mois peuvent être donnés sur un ou deux

chiffres). Le type de données de la colonne est objet. Les dates sont encodées sous forme de chaînes de caractères.Affichez cette colonne.

6. Utilisez la fonction 'to_datetime' pour convertir la colonne en un objet datetime64. Cela rendra le format de la date cohérent sur toutes les lignes :

```
0        2006-09-16
1        2004-09-01
2        2003-11-01
3        2004-05-01
4        2004-09-13
            ...
11118    2004-12-21
11119    1988-12-01
```

7. Vous pouvez convertir une colonne contenant des dates lorsque vous chargez les données à partir du fichier CSV avec la méthode read_csv. Donnez le code.
8. Extraire le jour, le mois et l'année de la colonne publication_date et les stocker dans de nouvelles colonnes appelées pub_day, pub_month et pub_year .
9. Calculez le nombre de jours entre la première et la dernière publication d'un roman de Harry Potter.
10. Écrivez un code pour assigner à la variable 'potter_df' un DataFrame avec les lignes correspondant aux romans de Harry Potter.
11. Calculez la différence entre la date la plus récente et la date la plus ancienne des romans de Harry Potter.
12. Reformatez les dates de la colonne ‘publication_date’ pour qu'elles apparaissent au format "jour/mois/année" (par exemple, "01/12/2023").
13. Testez et interprétez le code suivant :

```
import sqlite3

conn = sqlite3.connect("./data/books.db")
```

14. Sélectionner un DataFrame contenant les colonnes ‘bookID, title, language_code, publisher et pub_year’ du DataFrame books_df.

15. Importer le DataFrame sélectionné dans une nouvelle table appelée ‘book’ dans la base de données SQLite.
16. Examinez la colonne ‘authors’ (si un livre a plusieurs auteurs, leurs noms sont séparés par des barres obliques /).
17. Écrivez un code pour créer une nouvelle table 'book_authors' dans la base de données. Dans votre code, vous devez :

    –>Obtenir à partir de books_df un DataFrame avec deux colonnes 'bookID' et 'author_name', où chaque ligne contient exactement un identifiant de livre et un nom d'auteur.

    –> Importer le DataFrame obtenu dans la base de données relationnelle.

Exercice 4 :

1. Créez une base de données SQLite appelée asc.db conformément aux spécifications définies dans le fichier ’./data/modele_logique.md’
2. Lire les deux fichiers ‘student_memberships.csv’ et ‘student_registrations.csv’ dans deux DataFrames séparés.
3. Obtenir à partir de ces deux DataFrames les DataFrames correspondant aux tables de votre modèle physique.
4. Écrivez un code pour supprimer les lignes en double de chaque DataFrame.
5. Vérifier que vous avez : 602 étudiants, 11 associations, 602 adresses électroniques différentes, 3 éditions, 900 inscriptions et 895 adhésions à des associations.
6. Les informations relatives au sexe ne sont pas cohérentes entre les lignes. Les références M, H ou garçon sont utilisées pour désigner les hommes. Les références F, W ou fille sont utilisées pour désigner les femmes. Nous voulons que les hommes soient toujours référencés comme M et que les femmes soient toujours référencées comme F.

   Écrivez le code pour normaliser les informations relatives au sexe dans le DataFrame.

7. Écrivez des assertions pour vérifier que les informations relatives au sexe sont maintenant cohérentes.

8. Les dates d'inscription et de paiement sont au format AAAA-MM-JJ. Nous voulons qu'elles soient exprimées sous la forme JJ-MM-AAAA.

   Écrivez le code pour changer le format des dates d'inscription et de paiement.

9. Écrivez le code pour importer les données dans la base de données asc.db

Bon travail