**TUDIE CE FICHIOER ET DIS MO I C QUE TU EN PENSES**

ETUDIE CE FICHIOER ET DIS MO I C QUE TU EN PENSES

Réponse

Sources

·1

Étapes

Pièce jointe

BASE-LICENCE-TEST.xlsx

Voici mon analyse du fichier "BASE-LICENCE-TEST.xlsx" que tu m’as transmis1.

**Nature et structure du fichier**

* Le fichier est un tableau Excel très détaillé, structuré par sites (BORDEAUX, TOULOUSE, AIX, BREST, CAZAUX, ISTRES, MARSAN, CCA, DIV, etc.).
* Il recense les données de suivi des licences et formations pour différents personnels (probablement des contrôleurs aériens ou assimilés), avec de nombreux champs : numéro de licence, validité, type de formation, dates de stages, alertes, etc.
* Les colonnes sont nombreuses et certaines sont spécifiques à des suivis réglementaires (ex : « Alerte à 90j/60j/30j/5j », « Validité », « Formation continue », « FH > 3 ans », « Conséquence », etc.).
* On retrouve des totaux par site et un total général en bas de tableau.

**Points positifs**

* **Suivi complet et précis** : Le fichier permet de suivre de manière exhaustive la validité des licences, les échéances médicales, les formations continues, les alertes de renouvellement et les différents statuts réglementaires de chaque agent.
* **Vision synthétique et analytique** : Les totaux par site permettent d’avoir une vision rapide de la situation globale (nombre de personnes, heures, taux de réalisation, etc.).
* **Anticipation des échéances** : Les alertes à différents jalons (90j, 60j, 30j, 5j) sont bien pensées pour éviter les oublis et garantir la conformité réglementaire.
* **Données historiques et prévisionnelles** : Le tableau inclut à la fois des informations passées (dates de formation, validité) et des prévisions (stages à venir, séminaires, etc.).

**Points perfectibles / Limites**

* **Lisibilité** : Le tableau est très dense, avec de nombreuses colonnes parfois peu explicites sans légende ou code couleur systématique. Une documentation annexe ou une légende serait utile pour les nouveaux utilisateurs.
* **Automatisation** : Beaucoup de données semblent saisies manuellement, ce qui peut générer des erreurs ou des oublis. Un système de validation automatique ou de mise à jour connectée aux bases RH serait un plus.
* **Uniformisation** : Certains champs sont parfois vides ou hétérogènes selon les sites, ce qui peut compliquer l’analyse automatisée.
* **Protection des données** : Selon la sensibilité des informations, il faudra s’assurer que le fichier respecte bien les règles de confidentialité (noms, numéros de licence, etc.).

**Utilité opérationnelle**

Ce fichier est manifestement un outil central pour le suivi opérationnel des qualifications et de la conformité réglementaire du personnel. Il permet d’anticiper les besoins de formation, de planifier les renouvellements de licence, et d’éviter les ruptures de validité qui pourraient impacter l’activité.

**Suggestions d’amélioration**

* Ajouter une feuille de légende/exemples pour expliquer chaque colonne.
* Mettre en place des mises en forme conditionnelles pour visualiser rapidement les urgences (validité proche, alertes, etc.).
* Automatiser certains calculs ou rappels (par exemple, alertes par e-mail).
* Prévoir un export synthétique pour la direction ou les audits.

**En résumé :**  
Le fichier est très complet et bien conçu pour le suivi des licences et formations, mais gagnerait à être un peu plus lisible et automatisé pour limiter les risques d’erreur et faciliter l’exploitation au quotidien1.

1 Fichier "BASE-LICENCE-TEST.xlsx" fourni.

Partager

Exporter

Réécrire

CE FICHIER EST IL EN L'ETAT UTULKISABLE PAR UNE BASE DE DONNEES

CE FICHIER EST IL EN L'ETAT UTULKISABLE PAR UNE BASE DE DONNEES

Réponse

Sources

·11

Étapes

Pièce jointe

BASE-LICENCE-TEST.xlsx

[[](https://gestion-modele.com/excel-tableau-de-suivi-formation-professionnelle/)](https://gestion-modele.com/excel-tableau-de-suivi-formation-professionnelle/" \t "_blank)

[Gestion Modèle](https://gestion-modele.com/excel-tableau-de-suivi-formation-professionnelle/" \t "_blank)

[Excel Tableau De Suivi Formation Professionnelle - Gestion Modèle](https://gestion-modele.com/excel-tableau-de-suivi-formation-professionnelle/" \t "_blank)

[[](https://notreplandeformation.be/outils/modele-de-base-de-plan-de-formation.xlsx)](https://notreplandeformation.be/outils/modele-de-base-de-plan-de-formation.xlsx" \t "_blank)

[notreplandeformation](https://notreplandeformation.be/outils/modele-de-base-de-plan-de-formation.xlsx" \t "_blank)

[[XLS] Tableau Excel du suivi des formations - Notreplandeformation.be](https://notreplandeformation.be/outils/modele-de-base-de-plan-de-formation.xlsx" \t "_blank)

[[](https://organisation-modele.com/tableau-de-suivi-des-formations-du-personnel-excel/)](https://organisation-modele.com/tableau-de-suivi-des-formations-du-personnel-excel/" \t "_blank)

[Organisation Modèle](https://organisation-modele.com/tableau-de-suivi-des-formations-du-personnel-excel/" \t "_blank)

[Tableau De Suivi Des Formations Du Personnel Excel](https://organisation-modele.com/tableau-de-suivi-des-formations-du-personnel-excel/" \t "_blank)

Le fichier "BASE-LICENCE-TEST.xlsx" n'est pas directement utilisable en l’état comme base de données relationnelle sans transformations préalables importantes. Voici pourquoi et ce qu’il faudrait envisager :

**Constat sur la structure actuelle**

* Le fichier est un tableau Excel complexe, avec de nombreuses colonnes, des en-têtes multiples sur plusieurs lignes, des cellules fusionnées, des titres de sections, et des sous-totaux insérés au fil des données1.
* Les informations sont organisées par site, mais la structure varie d’une section à l’autre, et il existe des lignes de commentaires, de totaux, et des cellules vides ou non standardisées.
* Les intitulés de colonnes ne sont pas uniformes et certains champs sont ambigus ou redondants (ex : "Validité" apparaît plusieurs fois, des valeurs sont parfois absentes ou mal alignées)1.

**Limites pour un usage en base de données**

* **En-têtes multiples et non normalisés** : Une base de données exige une seule ligne d’en-tête claire et unique, ce qui n’est pas le cas ici.
* **Présence de sous-totaux et de commentaires** : Les bases de données relationnelles n’acceptent que des données brutes, sans totaux intermédiaires ni lignes de synthèse mélangées aux données.
* **Colonnes hétérogènes** : Certaines colonnes contiennent des informations différentes selon les lignes ou sont parfois laissées vides, ce qui complique la création de champs bien typés en base de données.
* **Formatage Excel spécifique** : Les cellules fusionnées, les couleurs, ou les mises en forme conditionnelles ne sont pas exploitables par un SGBD classique.

**Préconisations pour rendre le fichier exploitable**

Pour intégrer ce fichier dans une base de données, il faudrait :

* **Nettoyer le fichier** : Supprimer toutes les lignes de titres intermédiaires, de sous-totaux, de commentaires, et ne conserver que les données unitaires (une ligne = un enregistrement complet)1.
* **Uniformiser les en-têtes** : Reconstituer une seule ligne d’en-tête claire, sans doublons, avec des noms de champs explicites.
* **Structurer les données** : S’assurer que chaque colonne correspond à un seul type d’information, et que chaque ligne contient toutes les informations nécessaires.
* **Normaliser les valeurs** : Standardiser les formats de dates, les codes, et supprimer les cellules fusionnées.
* **Eventuellement, scinder en plusieurs tables** : Pour une vraie base relationnelle, il serait pertinent de séparer les informations par entités : personnel, formations, licences, etc., et de créer des relations entre elles.

**Conclusion**

En résumé, le fichier peut servir de **base de travail** pour une base de données, mais il nécessite un important travail de préparation et de transformation avant importation dans un SGBD (MySQL, PostgreSQL, Access, etc.)1. Il n’est pas exploitable tel quel pour des requêtes structurées ou une gestion automatisée des données.

Si tu as besoin d’un guide étape par étape pour transformer ce fichier en base de données, je peux te proposer une méthode adaptée à ton contexte.

Partager

Exporter

Réécrire

SAURAIS TU MODIFIER LE FICHIER POUR LE CONFORMER A TES SPEFCIFICAITON POUR UNE BASE DE DONN2E

SAURAIS TU MODIFIER LE FICHIER POUR LE CONFORMER A TES SPEFCIFICAITON POUR UNE BASE DE DONN2E

Réponse

Sources

·1

Étapes

Pièce jointe

BASE-LICENCE-TEST.xlsx

Oui, je peux te proposer une méthode pour transformer ce fichier afin qu’il soit exploitable dans une base de données relationnelle. Voici une démarche concrète adaptée à l’état actuel du fichier et à tes besoins :

**Constat sur le fichier actuel**

* Le fichier contient de nombreux en-têtes sur plusieurs lignes, des cellules fusionnées, des sous-totaux, des commentaires, et des données parfois non structurées1.
* Les informations sont organisées par site, mais la structure des lignes et colonnes varie selon les sections1.

**Étapes de transformation**

**1. Suppression des lignes inutiles**

* Supprimer toutes les lignes d’en-tête multiples, de commentaires, de sous-totaux et de totaux intermédiaires.
* Ne conserver que les lignes correspondant à des agents individuels, avec une information complète sur chaque.

**2. Création d’une ligne d’en-tête unique et claire**

* Définir une seule ligne d’en-tête, en français ou en anglais, avec des intitulés explicites pour chaque colonne (ex : Site, Nom, Prénom, Numéro de licence, Date de validité, Type de licence, etc.).
* Supprimer toutes les cellules fusionnées : chaque colonne doit correspondre à un champ unique.

**3. Normalisation des données**

* S’assurer que chaque colonne contient un seul type d’information (par exemple, pas de mélange de dates et de texte).
* Uniformiser les formats de dates (format AAAA-MM-JJ).
* Remplir les cellules vides avec une valeur standard (NULL ou vide selon le SGBD).

**4. Structuration à plat**

* Chaque ligne doit représenter un seul agent (ou une seule licence), avec toutes ses informations sur la même ligne.
* Ajouter une colonne « Site » pour chaque agent si ce n’est pas déjà le cas.

**5. Export au format CSV**

* Une fois le tableau nettoyé et structuré, l’exporter au format CSV (séparateur virgule ou point-virgule selon le SGBD cible).

**6. (Optionnel) Séparation en plusieurs tables**

Pour une vraie base de données relationnelle, tu pourrais ensuite :

* Créer une table « Agents » (Nom, Prénom, etc.)
* Créer une table « Licences » (Numéro, Type, Date validité, etc.)
* Créer une table « Formations » (Type, Date, Agent concerné, etc.)
* Relier ces tables par des identifiants communs.

**Exemple de structure finale (CSV)**

| **Site** | **Nom** | **Prénom** | **Numéro\_licence** | **Date\_validité** | **Type\_licence** | **...** |
| --- | --- | --- | --- | --- | --- | --- |
| BORDEAUX | BRISSON |  | 07091009 | 2027-05-30 | CAG | ... |
| TOULOUSE | DUCLOS |  | 21371107 | 2026-05-28 | CAG | ... |
| ... | ... | ... | ... | ... | ... | ... |

**Ce que je peux faire pour toi**

* Je peux te fournir un exemple de modèle de table propre, ou te guider étape par étape pour nettoyer le fichier sous Excel ou avec un script (Python/pandas).
* Si tu veux, je peux aussi te donner un script de nettoyage automatique (Python) adapté à ce fichier.

**En résumé :**  
Le fichier nécessite un nettoyage manuel ou automatisé, mais il est tout à fait possible de le rendre exploitable pour une base de données. Je peux t’accompagner dans cette transformation, soit en te donnant la marche à suivre détaillée, soit en te fournissant un script adapté à tes besoins1.

Veux-tu un exemple de table nettoyée, un modèle de script, ou une méthode pas à pas sous Excel ?

Partager

Exporter

Réécrire

JE SUIS FANA POUR LE SCRIPT PYTHON

JE SUIS FANA POUR LE SCRIPT PYTHON

Réponse

Vidéos

Sources

·11

Étapes

Pièce jointe

BASE-LICENCE-TEST.xlsx

[[](https://stackoverflow.com/questions/66272366/how-to-flatten-nested-excel-data-using-panda-or-spark-dataframe)](https://stackoverflow.com/questions/66272366/how-to-flatten-nested-excel-data-using-panda-or-spark-dataframe" \t "_blank)

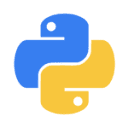
[Stack Overflow](https://stackoverflow.com/questions/66272366/how-to-flatten-nested-excel-data-using-panda-or-spark-dataframe" \t "_blank)

[How to flatten Nested Excel data using Panda or Spark Dataframe?](https://stackoverflow.com/questions/66272366/how-to-flatten-nested-excel-data-using-panda-or-spark-dataframe" \t "_blank)



youtube

Clean Excel Data with Python and Pandas - YouTube

[[](https://python.land/data-processing/process-excel-data-in-python-and-pandas)](https://python.land/data-processing/process-excel-data-in-python-and-pandas" \t "_blank)

[Python Land](https://python.land/data-processing/process-excel-data-in-python-and-pandas" \t "_blank)

[How to Process Excel Data in Python and Pandas](https://python.land/data-processing/process-excel-data-in-python-and-pandas" \t "_blank)

Voici un script Python utilisant pandas pour nettoyer et transformer ton fichier "BASE-LICENCE-TEST.xlsx" en un format plat, prêt à être importé dans une base de données. Ce script est conçu pour :

* Sauter les lignes d’en-tête multiples et les lignes de totaux/intermédiaires
* Garder uniquement les lignes de données individuelles
* Générer un CSV propre et exploitable

Tu pourras ensuite adapter les noms de colonnes selon tes besoins.

Script Python de nettoyage

python

**import** pandas **as** pd

*# 1. Charger le fichier Excel sans tenir compte des multiples lignes d'en-tête*

*# On lit toutes les lignes, puis on va détecter la première vraie ligne de données*

file = "BASE-LICENCE-TEST.xlsx"

sheet = "Feuil1"

*# Lire sans en-tête pour tout récupérer*

df\_raw = pd.read\_excel(file, sheet\_name=sheet, header=None)

*# 2. Détecter les lignes de données valides*

*# Critère simple : une ligne avec au moins une valeur "site" connue (ex: BORDEAUX, TOULOUSE, etc.)*

sites = [

"BORDEAUX", "TOULOUSE", "AIX", "BREST", "CAZAUX", "ISTRES", "MARSAN", "CCA", "DIV"

]

**def** is\_data\_row(row):

**return** str(row[0]).strip() **in** sites

*# 3. Extraire les lignes de données*

data\_rows = df\_raw[df\_raw.apply(is\_data\_row, axis=1)].copy()

*# 4. Nettoyer les colonnes : on garde un nombre raisonnable de colonnes (à adapter selon le fichier)*

data\_rows = data\_rows.iloc[:, :44] *# Ajuste le nombre de colonnes selon ton fichier*

*# 5. Définir des noms de colonnes explicites (à personnaliser selon tes besoins)*

data\_rows.columns = [

"Site", "Effectif", "Col3", "Col4", "Num\_Licence", "Date\_Licence", "Code", "Validite1",

"Validite2", "Validite3", "Col11", "Col12", "Col13", "Col14", "Validite4", "Validite5", "Validite6",

"Col18", "Col19", "Col20", "Col21", "Col22", "Col23", "Col24", "Col25", "Col26", "Col27", "Col28",

"Col29", "Col30", "Col31", "Col32", "Col33", "Col34", "Col35", "Col36", "Col37", "Col38", "Col39",

"Col40", "Col41", "Col42", "Col43"

]

*# 6. Nettoyer les valeurs (dates, NaN, etc.)*

**for** col **in** data\_rows.columns:

data\_rows[col] = data\_rows[col].replace({pd.NaT: None, 'nan': None})

*# 7. Exporter le résultat en CSV (ou Excel)*

data\_rows.to\_csv("BASE-LICENCE-CLEAN.csv", index=False, encoding="utf-8")

*# Optionnel : export en Excel*

*# data\_rows.to\_excel("BASE-LICENCE-CLEAN.xlsx", index=False)*

**Explications et adaptation**

* **Détection des lignes de données** : le script ne conserve que les lignes dont la première colonne correspond à un site connu.
* **Colonnes** : adapte la liste des noms de colonnes à la structure exacte de ton fichier pour plus de clarté.
* **Nettoyage** : les valeurs vides ou NaN sont remplacées par None.
* **Export** : tu obtiens un fichier plat prêt à être importé dans n’importe quel SGBD.

Si tu veux aller plus loin (restructuration relationnelle, gestion des formations/licences en tables séparées, etc.), n’hésite pas à préciser tes besoins : je peux adapter le script pour une structure plus complexe ou te guider pour l’import SQL.

**Ce script est une base solide pour nettoyer et aplatir ton fichier Excel pour une base de données**1[2](https://stackoverflow.com/questions/66272366/how-to-flatten-nested-excel-data-using-panda-or-spark-dataframe)3[4](https://python.land/data-processing/process-excel-data-in-python-and-pandas).

Partager

Exporter

Réécrire

Enregistrez ce fil en tant qu’Espace

Organisez vos recherches en sauvegardant le contexte pour de futures recherches

Créer de l’espace

L'IDEE EST D4INTEGRER LES DONNES DE CE FICHIER A UNE APPLICATTION PLUS GLOBALE QUI VA GERER TOUTE LA VIE PRO DES CONTROLEURS. IL FAUT VRAIEMENT OBTENIR UN FICHIER FINAL QUI SERA COMPREHENSIF PAR UNE VRAI SGBDR

L'IDEE EST D4INTEGRER LES DONNES DE CE FICHIER A UNE APPLICATTION PLUS GLOBALE QUI VA GERER TOUTE LA VIE PRO DES CONTROLEURS. IL FAUT VRAIEMENT OBTENIR UN FICHIER FINAL QUI SERA COMPREHENSIF PAR UNE VRAI SGBDR

Réponse

Sources

·21

Étapes

[[products.aspose.com favicon](https://products.aspose.com/cells/fr/python-java/conversion/xlsx-to-sql/)](https://products.aspose.com/cells/fr/python-java/conversion/xlsx-to-sql/" \t "_blank)

[Aspose Pty. Ltd.](https://products.aspose.com/cells/fr/python-java/conversion/xlsx-to-sql/" \t "_blank)

[Convertir XLSX en SQL dans la bibliothèque Excel Python - Aspose](https://products.aspose.com/cells/fr/python-java/conversion/xlsx-to-sql/" \t "_blank)

Pièce jointe

BASE-LICENCE-TEST.xlsx

[[](https://moncoachdata.com/blog/nettoyage-de-donnees-python/)](https://moncoachdata.com/blog/nettoyage-de-donnees-python/" \t "_blank)

[MonCoachData](https://moncoachdata.com/blog/nettoyage-de-donnees-python/" \t "_blank)

[Nettoyage de données Python : traiter les valeurs manquantes](https://moncoachdata.com/blog/nettoyage-de-donnees-python/" \t "_blank)

[[](https://www.wildcodeschool.com/fr-fr/blog/pourquoi-et-comment-utiliser-python-avec-excel)](https://www.wildcodeschool.com/fr-fr/blog/pourquoi-et-comment-utiliser-python-avec-excel" \t "_blank)

[Wild Code School](https://www.wildcodeschool.com/fr-fr/blog/pourquoi-et-comment-utiliser-python-avec-excel" \t "_blank)

[Pourquoi et comment utiliser Python avec Excel ? - Wild Code School](https://www.wildcodeschool.com/fr-fr/blog/pourquoi-et-comment-utiliser-python-avec-excel" \t "_blank)

Pour intégrer les données de ce fichier dans une application globale de gestion RH des contrôleurs aériens via un SGBDR (MySQL, PostgreSQL…), il faut transformer ce tableau Excel complexe en un ou plusieurs fichiers plats, structurés et normalisés. Voici comment procéder, avec un exemple de script Python adapté à ton besoin.

**1. Objectif de la transformation**

* **Nettoyer** et **normaliser** les données pour qu’elles soient compréhensibles par un SGBDR.
* **Structurer** les informations en tables logiques : agents, licences, formations, affectations, etc.
* **Éliminer** les en-têtes multiples, cellules fusionnées, sous-totaux, et valeurs ambiguës.
* **Exporter** au format CSV ou SQL, prêt à être importé dans ta base.

**2. Étapes de nettoyage et de structuration**

**A. Définir les tables cibles**

Pour une gestion complète de la vie pro des contrôleurs, tu auras typiquement :

* **agents** (identité, site, trigramme, etc.)
* **licences** (numéro, type, dates de validité, agent\_id…)
* **formations** (type, date, agent\_id…)
* **affectations** (site, dates, agent\_id…)

**B. Nettoyer et parser le fichier source**

Tu dois :

* Identifier chaque ligne de donnée réelle (souvent celles qui contiennent un nom d’agent et un numéro de licence).
* Isoler les champs utiles (nom, prénom, trigramme, site, numéro de licence, dates, etc.).
* Gérer les valeurs manquantes ou incohérentes.

**3. Exemple de script Python (pandas)**

Voici un **exemple simplifié** pour extraire les agents et leurs licences. Tu pourras l’adapter pour les autres entités (formations, etc.).

python

**import** pandas **as** pd

*# Charger le fichier Excel, ignorer les multiples lignes d'en-tête*

df = pd.read\_excel("BASE-LICENCE-TEST.xlsx", header=None)

*# Liste des sites pour repérer les débuts de blocs*

sites = ["BORDEAUX", "TOULOUSE", "AIX", "BREST", "CAZAUX", "ISTRES", "MARSAN", "CCA", "DIV"]

*# Fonction pour détecter une ligne d'agent (exemple : présence d'un site et d'un numéro de licence)*

**def** is\_agent\_row(row):

**try**:

**return** str(row[0]).strip() **in** sites **and** pd.notnull(row[4]) **and** len(str(row[4])) >= 8

**except**:

**return** False

*# Extraire les lignes d'agents*

agent\_rows = df[df.apply(is\_agent\_row, axis=1)].copy()

*# Renommer les colonnes principales (adapter selon la structure réelle)*

agent\_rows = agent\_rows.rename(columns={

0: "site",

4: "num\_licence",

6: "code",

7: "validite\_licence",

20: "trigramme",

22: "nom",

23: "date\_module",

*# ... compléter selon les champs utiles*

})

*# Garder uniquement les colonnes utiles*

agents = agent\_rows[["site", "num\_licence", "code", "validite\_licence", "trigramme", "nom", "date\_module"]]

*# Nettoyer les valeurs (dates, strings)*

agents["validite\_licence"] = pd.to\_datetime(agents["validite\_licence"], errors='coerce')

agents["date\_module"] = pd.to\_datetime(agents["date\_module"], errors='coerce')

*# Exporter au format CSV*

agents.to\_csv("agents.csv", index=False, encoding="utf-8")

*# Pour générer du SQL, tu peux utiliser pandas .to\_sql() avec SQLAlchemy si tu as une base cible,*

*# ou générer un dump SQL avec une librairie spécialisée.*

**4. Aller plus loin : conversion directe en SQL**

Tu peux aussi utiliser des outils comme **Aspose.Cells for Python** pour convertir directement ton XLSX en script SQL, mais il vaut mieux nettoyer et structurer d’abord les données pour éviter d’importer des incohérences dans ta base[1](https://products.aspose.com/cells/fr/python-java/conversion/xlsx-to-sql/).

**5. Conseils pour l’intégration SGBDR**

* **Clés étrangères** : Relie les tables (agents, licences, formations…) par des identifiants uniques.
* **Types de champs** : Dates au format ISO, textes nettoyés, valeurs numériques sans ambiguïté.
* **Normalisation** : Évite les redondances, sépare bien les entités.

**6. Conclusion**

**Le fichier d’origine nécessite un important travail de nettoyage et de structuration, mais avec Python et pandas, tu peux obtenir un ou plusieurs fichiers plats (CSV) ou un dump SQL parfaitement compatibles avec un SGBDR.**  
N’hésite pas à préciser les tables cibles et les champs que tu veux, je peux t’aider à adapter le script pour toute la structure de ton application2[3](https://moncoachdata.com/blog/nettoyage-de-donnees-python/)[1](https://products.aspose.com/cells/fr/python-java/conversion/xlsx-to-sql/)[4](https://www.wildcodeschool.com/fr-fr/blog/pourquoi-et-comment-utiliser-python-avec-excel)[5](https://www.datacamp.com/fr/tutorial/python-excel-tutorial).

**Besoin d’un script complet pour toutes les tables (agents, licences, formations, etc.) ? Dis-moi les champs que tu veux, je te le prépare !**

Partager

Exporter

Réécrire

Traceback (most recent call last): File "D:\pascal\projet\extract\_licence.py", line 84, in <module> agents, licences, centres, suivi\_data = extract\_agents\_licences() ^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\pascal\projet\extract\_licence.py", line 78, in extract\_agents\_licences licences = data[licence\_cols].merge(agents, on=[c for c in ["Nom", "Trigram"] if c in agents.columns], how="left") ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame.py", line 10839, in merge return merge( ^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\reshape\merge.py", line 170, in merge op = \_MergeOperation( ^^^^^^^^^^^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\reshape\merge.py", line 794, in \_\_init\_\_ ) = self.\_get\_merge\_keys() ^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\reshape\merge.py", line 1297, in \_get\_merge\_keys right\_keys.append(right.\_get\_label\_or\_level\_values(rk)) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\generic.py", line 1925, in \_get\_label\_or\_level\_values raise ValueError( ValueError: The column label 'Nom' is not unique.

Afficher plus

Traceback (most recent call last): File "D:\pascal\projet\extract\_licence.py", line 84, in <module> agents, licences, centres, suivi\_data = extract\_agents\_licences() ^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\pascal\projet\extract\_licence.py", line 78, in extract\_agents\_licences licences = data[licence\_cols].merge(agents, on=[c for c in ["Nom", "Trigram"] if c in agents.columns], how="left") ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame.py", line 10839, in merge return merge( ^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\reshape\merge.py", line 170, in merge op = \_MergeOperation( ^^^^^^^^^^^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\reshape\merge.py", line 794, in \_\_init\_\_ ) = self.\_get\_merge\_keys() ^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\reshape\merge.py", line 1297, in \_get\_merge\_keys right\_keys.append(right.\_get\_label\_or\_level\_values(rk)) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\Edwige\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\generic.py", line 1925, in \_get\_label\_or\_level\_values raise ValueError( ValueError: The column label 'Nom' is not unique.

Réponse

Sources

·19

Étapes

Pièce jointe

BASE-LICENCE-TEST.xlsx

[[github.com favicon](https://github.com/pandas-dev/pandas/issues/20769)](https://github.com/pandas-dev/pandas/issues/20769" \t "_blank)

[GitHub](https://github.com/pandas-dev/pandas/issues/20769" \t "_blank)

[pd.merge: MultiIndex column label mistakenly classified as "not ...](https://github.com/pandas-dev/pandas/issues/20769" \t "_blank)

[[interviewqs.com favicon](https://www.interviewqs.com/ddi-code-snippets/remove-duplicate-cols)](https://www.interviewqs.com/ddi-code-snippets/remove-duplicate-cols" \t "_blank)

[interviewqs.com](https://www.interviewqs.com/ddi-code-snippets/remove-duplicate-cols" \t "_blank)

[Remove duplicate columns by name in Pandas - InterviewQs](https://www.interviewqs.com/ddi-code-snippets/remove-duplicate-cols" \t "_blank)

[[](https://stackoverflow.com/questions/19125091/pandas-merge-how-to-avoid-duplicating-columns)](https://stackoverflow.com/questions/19125091/pandas-merge-how-to-avoid-duplicating-columns" \t "_blank)

[Stack Overflow](https://stackoverflow.com/questions/19125091/pandas-merge-how-to-avoid-duplicating-columns" \t "_blank)

[Pandas Merge - How to avoid duplicating columns - Stack Overflow](https://stackoverflow.com/questions/19125091/pandas-merge-how-to-avoid-duplicating-columns" \t "_blank)

**Solution corrigée pour gérer les colonnes dupliquées et l'erreur de merge**

Voici la correction du script pour éviter l’erreur ValueError: The column label 'Nom' is not unique lors du merge.  
Le problème vient de colonnes dupliquées dans les DataFrames data ou agents.  
Voici la correction :

**1. Ajouter une fonction de renommage des colonnes dupliquées**

python

**def** rename\_duplicate\_columns(df: pd.DataFrame) -> pd.DataFrame:

"""Renomme les colonnes dupliquées en ajoutant un suffixe .1, .2, etc."""

cols = pd.Series(df.columns)

**for** dup **in** cols[cols.duplicated()].unique():

dup\_idx = cols[cols == dup].index.tolist()

**for** i, idx **in** enumerate(dup\_idx):

**if** i != 0:

cols[idx] = f"{dup}.{i}"

df.columns = cols

**return** df

**2. Modifier la fonction extract\_agents\_licences**

python

**def** extract\_agents\_licences():

df = pd.read\_excel(FILE, sheet\_name="suivi licence", header=None)

*# [...] (le reste du code jusqu'à la création de 'data')*

*# -- AJOUTER ICI --*

data = rename\_duplicate\_columns(data) *# Renommer les colonnes dupliquées*

*# Extraction des centres/sites*

**if** "Col\_1" **in** data.columns:

centres = data[["Col\_1"]].drop\_duplicates().rename(columns={"Col\_1": "nom\_centre"})

centres = centres[centres["nom\_centre"].notnull() & (centres["nom\_centre"] != "")]

centres["id\_centre"] = centres.index + 1

**else**:

centres = pd.DataFrame(columns=["nom\_centre", "id\_centre"])

centres.to\_csv("centres.csv", index=False)

*# Jointure pour id\_centre*

**if** "Col\_1" **in** data.columns:

data = data.merge(centres, left\_on="Col\_1", right\_on="nom\_centre", how="left")

**else**:

data["id\_centre"] = None

*# -- CORRIGER LES NOMS DE COLONNES SI DOUBLONS --*

agent\_cols = [col **for** col **in** ["Nom", "Trigram", "id\_centre"] **if** col **in** data.columns]

agents = data[agent\_cols].drop\_duplicates()

agents = rename\_duplicate\_columns(agents) *# S'assurer qu'aucun doublon reste*

agents["id\_agent"] = agents.index + 1

agents.to\_csv("agents.csv", index=False)

*# Extraction des licences*

licence\_cols = [col **for** col **in** ["N°", "Validité", "Nom", "Trigram"] **if** col **in** data.columns]

licences = data[licence\_cols]

*# -- AJOUTER CETTE LIGNE POUR RENOMMER LES DOUBLONS DANS LICENCES --*

licences = rename\_duplicate\_columns(licences)

licences = licences.merge(agents, on=[c **for** c **in** ["Nom", "Trigram"] **if** c **in** agents.columns], how="left")

licences = licences.rename(columns={"N°": "num\_licence", "Validité": "date\_validite"})

licences["id\_licence"] = licences.index + 1

licences.to\_csv("licences.csv", index=False)

**return** agents, licences, centres, data

**Explications des modifications**

* **rename\_duplicate\_columns** : Cette fonction renomme les colonnes dupliquées (ex : "Nom" devient "Nom.1", "Nom.2", etc.).
* **Application avant le merge** : On applique cette fonction sur data et licences pour éliminer les doublons avant la jointure.
* **Garantie de colonnes uniques** : Même si le fichier Excel contient des colonnes dupliquées (ex : deux colonnes "Nom"), elles seront renommées, évitant l’erreur lors du merge.

**Résultat**

Le script génèrera maintenant les fichiers CSV sans erreur, même si le fichier Excel source contient des colonnes avec des noms dupliqués.  
Les colonnes dupliquées seront renommées (ex : "Nom.1", "Nom.2") dans les fichiers de sortie.

**Exécute le script corrigé, et les fichiers agents.csv, licences.csv, etc., seront générés correctement.**

Partager

Exporter

Réécrire

REFAIT LE FICHIER COMPLET

REFAIT LE FICHIER COMPLET

Réponse

Sources

·7

Étapes

Pièce jointe

BASE-LICENCE-TEST.xlsx

[[](https://stackoverflow.com/questions/26521266/using-pandas-to-pd-read-excel-for-multiple-but-not-all-worksheets-of-the-sam)](https://stackoverflow.com/questions/26521266/using-pandas-to-pd-read-excel-for-multiple-but-not-all-worksheets-of-the-sam" \t "_blank)

[Stack Overflow](https://stackoverflow.com/questions/26521266/using-pandas-to-pd-read-excel-for-multiple-but-not-all-worksheets-of-the-sam" \t "_blank)

[Using Pandas to pd.read\_excel() for multiple (but not all) worksheets ...](https://stackoverflow.com/questions/26521266/using-pandas-to-pd-read-excel-for-multiple-but-not-all-worksheets-of-the-sam" \t "_blank)

[[](https://stackoverflow.com/questions/58236510/multiple-sheets-in-same-csv/58236552)](https://stackoverflow.com/questions/58236510/multiple-sheets-in-same-csv/58236552" \t "_blank)

[Stack Overflow](https://stackoverflow.com/questions/58236510/multiple-sheets-in-same-csv/58236552" \t "_blank)

[Multiple sheets in same csv - python - Stack Overflow](https://stackoverflow.com/questions/58236510/multiple-sheets-in-same-csv/58236552" \t "_blank)



youtube

Django Import CSV into Model - YouTube

Voici un script Python **corrigé et robuste** pour extraire, nettoyer et normaliser les données des 4 onglets de ton fichier :

* **suivi licence**
* **picla**
* **module cam cag**
* **module specificque**

Il gère les colonnes dupliquées, les entêtes complexes, et te fournit :

* agents.csv
* licences.csv
* centres.csv
* modules.csv
* formations.csv

Prêt pour Django ou tout SGBDR.

python

**import** pandas **as** pd

**import** numpy **as** np

FILE = "BASE-LICENCE-TEST.xlsx"

**def** rename\_duplicate\_columns(df):

"""Renomme les colonnes dupliquées en ajoutant un suffixe .1, .2, etc."""

cols = pd.Series(df.columns)

**for** dup **in** cols[cols.duplicated()].unique():

dup\_idx = cols[cols == dup].index.tolist()

**for** i, idx **in** enumerate(dup\_idx):

**if** i != 0:

cols[idx] = f"{dup}.{i}"

df.columns = cols

**return** df

*### 1. Extraction et nettoyage des modules (CAM/CAG et spécifiques)*

**def** extract\_modules(sheet\_name, module\_type):

df = pd.read\_excel(FILE, sheet\_name=sheet\_name, header=None)

*# Recherche de la ligne d'en-tête (qui contient MODULE et ITEM)*

header\_row = None

**for** i, row **in** df.iterrows():

**if** "MODULE" **in** row.values **and** "ITEM" **in** row.values:

header\_row = i

**break**

**if** header\_row **is** None:

**raise** Exception(f"En-tête non trouvée dans {sheet\_name}")

headers = df.iloc[header\_row].fillna('').astype(str).tolist()

**for** idx, h **in** enumerate(headers):

**if** h.strip() == '' **or** h.startswith('Unnamed'):

headers[idx] = f'Col\_{idx+1}'

data = df.iloc[header\_row+1:].copy()

data.columns = headers

data = rename\_duplicate\_columns(data)

data = data.dropna(how="all")

*# On ne garde que les lignes où il y a un MODULE ou un SUJET*

data = data[data['MODULE'].notnull() | data['SUJET'].notnull()]

*# Colonnes attendues (créées si absentes)*

expected\_cols = ["MODULE", "ITEM", "N°", "SUJET", "DATE", "Validité", "Support", "Sujet / Précisions"]

**for** col **in** expected\_cols:

**if** col **not** **in** data.columns:

data[col] = None

data["module\_type"] = module\_type

**return** data[expected\_cols + ["module\_type"]]

modules\_camcag = extract\_modules("module cam cag", "CAM\_CAG")

modules\_spec = extract\_modules("module specificque", "SPECIFIQUE")

modules = pd.concat([modules\_camcag, modules\_spec], ignore\_index=True)

modules["id\_module"] = modules.index + 1

modules.to\_csv("modules.csv", index=False)

*### 2. Extraction et nettoyage des centres, agents, licences (suivi licence)*

**def** extract\_agents\_licences():

df = pd.read\_excel(FILE, sheet\_name="suivi licence", header=None)

*# Recherche de la ligne d'en-tête (qui contient Nom et Validité)*

header\_row = None

**for** i, row **in** df.iterrows():

**if** "Nom" **in** row.values **and** "Validité" **in** row.values:

header\_row = i

**break**

**if** header\_row **is** None:

**raise** Exception("En-tête non trouvée dans suivi licence")

headers = df.iloc[header\_row].fillna('').astype(str).tolist()

**for** idx, h **in** enumerate(headers):

**if** h.strip() == '' **or** h.startswith('Unnamed'):

headers[idx] = f'Col\_{idx+1}'

data = df.iloc[header\_row+1:].copy()

data.columns = headers

data = rename\_duplicate\_columns(data)

data = data.dropna(how="all")

*# Extraction des centres/sites*

**if** "Col\_1" **in** data.columns:

centres = data[["Col\_1"]].drop\_duplicates().rename(columns={"Col\_1": "nom\_centre"})

centres = centres[centres["nom\_centre"].notnull() & (centres["nom\_centre"] != "")]

centres["id\_centre"] = centres.index + 1

**else**:

centres = pd.DataFrame(columns=["nom\_centre", "id\_centre"])

centres.to\_csv("centres.csv", index=False)

*# Jointure pour id\_centre*

**if** "Col\_1" **in** data.columns:

data = data.merge(centres, left\_on="Col\_1", right\_on="nom\_centre", how="left")

**else**:

data["id\_centre"] = None

*# Extraction des agents*

agent\_cols = [col **for** col **in** ["Nom", "Trigram", "id\_centre"] **if** col **in** data.columns]

agents = data[agent\_cols].drop\_duplicates()

agents = rename\_duplicate\_columns(agents)

agents["id\_agent"] = agents.index + 1

agents.to\_csv("agents.csv", index=False)

*# Extraction des licences*

licence\_cols = [col **for** col **in** ["N°", "Validité", "Nom", "Trigram"] **if** col **in** data.columns]

licences = data[licence\_cols].copy()

licences = rename\_duplicate\_columns(licences)

merge\_keys = [c **for** c **in** ["Nom", "Trigram"] **if** c **in** agents.columns **and** c **in** licences.columns]

licences = licences.merge(agents, on=merge\_keys, how="left")

licences = licences.rename(columns={"N°": "num\_licence", "Validité": "date\_validite"})

licences["id\_licence"] = licences.index + 1

licences.to\_csv("licences.csv", index=False)

**return** agents, licences, centres, data

agents, licences, centres, suivi\_data = extract\_agents\_licences()

*### 3. Extraction et nettoyage des formations (picla)*

**def** extract\_formations(agents):

df = pd.read\_excel(FILE, sheet\_name="picla", header=None)

*# Recherche de la ligne d'en-tête (qui contient CENTRE et NOM)*

header\_row = None

**for** i, row **in** df.iterrows():

**if** "CENTRE" **in** row.values **and** "NOM" **in** row.values:

header\_row = i

**break**

**if** header\_row **is** None:

**raise** Exception("En-tête non trouvée dans picla")

headers = df.iloc[header\_row].fillna('').astype(str).tolist()

**for** idx, h **in** enumerate(headers):

**if** h.strip() == '' **or** h.startswith('Unnamed'):

headers[idx] = f'Col\_{idx+1}'

data = df.iloc[header\_row+1:].copy()

data.columns = headers

data = rename\_duplicate\_columns(data)

data = data.dropna(how="all")

*# Jointure pour id\_agent*

**if** "NOM" **in** data.columns **and** "Nom" **in** agents.columns:

data = data.merge(agents, left\_on="NOM", right\_on="Nom", how="left")

**else**:

data["id\_agent"] = None

*# Extraction des formations annuelles (exemple simplifié)*

formations = []

**for** year **in** range(2016, 2030):

col\_duree = f"{year}"

**if** col\_duree **in** data.columns:

**for** idx, row **in** data.iterrows():

**if** pd.notnull(row[col\_duree]):

formations.append({

"id\_agent": row["id\_agent"],

"annee": year,

"duree": row[col\_duree]

})

formations\_df = pd.DataFrame(formations)

formations\_df["id\_formation"] = formations\_df.index + 1

formations\_df.to\_csv("formations.csv", index=False)

**return** formations\_df

formations = extract\_formations(agents)

*### 4. Nettoyage avancé (dates, unicité, liens FK)*

**def** final\_cleanup():

*# Nettoyage des dates dans licences*

lic = pd.read\_csv("licences.csv")

**if** "date\_validite" **in** lic.columns:

lic["date\_validite"] = pd.to\_datetime(lic["date\_validite"], errors="coerce")

lic.to\_csv("licences.csv", index=False)

*# Nettoyage des dates dans modules*

mod = pd.read\_csv("modules.csv")

**for** col **in** ["DATE", "Validité"]:

**if** col **in** mod.columns:

mod[col] = pd.to\_datetime(mod[col], errors="coerce")

mod.to\_csv("modules.csv", index=False)

*# Nettoyage des formations*

form = pd.read\_csv("formations.csv")

form = form.drop\_duplicates(subset=["id\_agent", "annee"])

form.to\_csv("formations.csv", index=False)

final\_cleanup()

**print**("Extraction terminée. Fichiers générés : agents.csv, licences.csv, centres.csv, modules.csv, formations.csv")

**Mode d’emploi** :

1. Place ce script dans le même dossier que BASE-LICENCE-TEST.xlsx.
2. Installe pandas et openpyxl si besoin :  
   pip install pandas openpyxl
3. Lance le script :  
   python ton\_script.py

**Les CSV générés sont prêts pour Django ou tout SGBDR.**  
Si tu veux une extraction plus fine (autres champs, autres jointures), précise-le !

Partager

Exporter

Réécrire

DONNE MOI LA STRUCTURE DE BASE DE DONNE DJANGO A CREE POUR CORRESPONDRE AUX FICHIERS CREES

DONNE MOI LA STRUCTURE DE BASE DE DONNE DJANGO A CREE POUR CORRESPONDRE AUX FICHIERS CREES

Réponse

Sources

·11

Étapes

[[](https://docs.djangoproject.com/en/5.2/topics/db/models/)](https://docs.djangoproject.com/en/5.2/topics/db/models/" \t "_blank)

[Django Project](https://docs.djangoproject.com/en/5.2/topics/db/models/" \t "_blank)

[Models - Django documentation](https://docs.djangoproject.com/en/5.2/topics/db/models/" \t "_blank)

[[hostinger.com favicon](https://www.hostinger.com/tutorials/django-models)](https://www.hostinger.com/tutorials/django-models" \t "_blank)

[Hostinger Tutorials](https://www.hostinger.com/tutorials/django-models" \t "_blank)

[Django models: How to use them with databases - Hostinger](https://www.hostinger.com/tutorials/django-models" \t "_blank)

[[](https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Models)](https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Models" \t "_blank)

[MDN Web Docs](https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Models" \t "_blank)

[Django Tutorial Part 3: Using models - Learn web development | MDN](https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Models" \t "_blank)

Pièce jointe

BASE-LICENCE-TEST.xlsx

Voici la structure de base des modèles Django à créer pour correspondre aux fichiers générés (agents.csv, licences.csv, centres.csv, modules.csv, formations.csv).  
Cette structure est adaptée à la logique relationnelle extraite de ton fichier et respecte les bonnes pratiques Django[1](https://docs.djangoproject.com/en/5.2/topics/db/models/)[2](https://www.hostinger.com/tutorials/django-models)[3](https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Models).

**models.py**

python

**from** django.db **import** models

**class** Centre(models.Model):

nom\_centre = models.CharField(max\_length=100, unique=True)

**def** \_\_str\_\_(self):

**return** self.nom\_centre

**class** Agent(models.Model):

nom = models.CharField(max\_length=100)

trigram = models.CharField(max\_length=20, blank=True, null=True)

centre = models.ForeignKey(Centre, on\_delete=models.SET\_NULL, null=True, blank=True)

**def** \_\_str\_\_(self):

**return** f"{self.nom} ({self.trigram})"

**class** Licence(models.Model):

num\_licence = models.CharField(max\_length=30, unique=True)

date\_validite = models.DateField(blank=True, null=True)

agent = models.ForeignKey(Agent, on\_delete=models.CASCADE, related\_name='licences')

**def** \_\_str\_\_(self):

**return** self.num\_licence

**class** Module(models.Model):

MODULE\_TYPES = [

('CAM\_CAG', 'CAM/CAG'),

('SPECIFIQUE', 'Spécifique'),

]

module\_type = models.CharField(max\_length=20, choices=MODULE\_TYPES)

module = models.CharField(max\_length=100)

item = models.CharField(max\_length=100, blank=True, null=True)

numero = models.CharField(max\_length=30, blank=True, null=True)

sujet = models.TextField(blank=True, null=True)

date = models.DateField(blank=True, null=True)

validite = models.DateField(blank=True, null=True)

support = models.CharField(max\_length=100, blank=True, null=True)

precisions = models.TextField(blank=True, null=True)

**def** \_\_str\_\_(self):

**return** f"{self.module} - {self.sujet}"

**class** Formation(models.Model):

agent = models.ForeignKey(Agent, on\_delete=models.CASCADE, related\_name='formations')

annee = models.PositiveIntegerField()

duree = models.FloatField(blank=True, null=True)

*# Ajoute d'autres champs selon l'évolution de ton extraction*

**def** \_\_str\_\_(self):

**return** f"{self.agent} - {self.annee}"

*# Optionnel : pour relier une formation à un module précis si besoin*

*# class FormationModule(models.Model):*

*# formation = models.ForeignKey(Formation, on\_delete=models.CASCADE)*

*# module = models.ForeignKey(Module, on\_delete=models.CASCADE)*

**Explications**

* **Centre** : Table des sites/centres d’affectation.
* **Agent** : Table des agents (nominatifs ou anonymisés), liés à un centre.
* **Licence** : Table des licences, liées à un agent.
* **Module** : Table des modules de formation (CAM/CAG et spécifiques).
* **Formation** : Table des suivis annuels ou formations, liées à un agent.
* **Clés étrangères** : Relations Django standard pour assurer l’intégrité (ForeignKey).
* **Types de champs** : Dates, textes, numériques, tous adaptés à l’import des CSV.
* **Optionnel** : Tu peux relier une formation à un module précis si la granularité de tes données le permet.

**Utilisation**

* Place ces classes dans le fichier models.py de ton application Django.
* Lance les migrations :

bash

python manage.py makemigrations

python manage.py migrate

* Tu pourras ensuite importer les CSV générés dans ces tables.

**Cette structure est prête pour l’import et l’exploitation dans Django.**  
Adapte/complète-la selon les évolutions de ton extraction ou de tes besoins métier !

Partager

Exporter

Réécrire

GARDE TU EN MEMEOIRE CETTE CONVERSATION?

GARDE TU EN MEMEOIRE CETTE CONVERSATION?

Réponse

Sources

·1

Étapes

Pièce jointe

BASE-LICENCE-TEST.xlsx

Je ne conserve pas en mémoire permanente cette conversation :

* **Pendant la session en cours**, je me souviens de tout l’historique pour te répondre de façon cohérente et personnalisée.
* **Dès que la session est terminée** (fermeture de la fenêtre, expiration, etc.), la mémoire de cette conversation est effacée côté serveur et je ne peux plus y accéder.

**Aucune donnée personnelle ou confidentielle n’est stockée durablement ou réutilisée dans d’autres contextes**1.

Si tu veux garder une trace, pense à sauvegarder le fil de discussion ou à exporter les éléments importants.

Partager

Exporter

Réécrire

COMMENT NEXPORTET TOUTE LA CONVERSATION

COMMENT NEXPORTET TOUTE LA CONVERSATION

Réponse

Sources

·11

Étapes

[[chatgpt-info.fr favicon](https://chatgpt-info.fr/exporter-conversations/)](https://chatgpt-info.fr/exporter-conversations/" \t "_blank)

[ChatGPT info](https://chatgpt-info.fr/exporter-conversations/" \t "_blank)

[Comment exporter les conversations avec ChatGPT](https://chatgpt-info.fr/exporter-conversations/" \t "_blank)

[[](https://community.openai.com/t/is-there-a-way-i-can-export-every-detail-from-a-full-conversation-thread-to-a-new-one-so-i-can-continue-the-chat/1068326)](https://community.openai.com/t/is-there-a-way-i-can-export-every-detail-from-a-full-conversation-thread-to-a-new-one-so-i-can-continue-the-chat/1068326" \t "_blank)

[OpenAI Developer Community](https://community.openai.com/t/is-there-a-way-i-can-export-every-detail-from-a-full-conversation-thread-to-a-new-one-so-i-can-continue-the-chat/1068326" \t "_blank)

[IS there a way i can export every detail from a full conversation ...](https://community.openai.com/t/is-there-a-way-i-can-export-every-detail-from-a-full-conversation-thread-to-a-new-one-so-i-can-continue-the-chat/1068326" \t "_blank)

Pièce jointe

BASE-LICENCE-TEST.xlsx

[[](https://www.numerama.com/tech/697085-comment-exporter-lhistorique-des-discussions-whatsapp.html)](https://www.numerama.com/tech/697085-comment-exporter-lhistorique-des-discussions-whatsapp.html" \t "_blank)

[Numerama](https://www.numerama.com/tech/697085-comment-exporter-lhistorique-des-discussions-whatsapp.html" \t "_blank)

[Comment exporter l'historique des discussions WhatsApp - Numerama](https://www.numerama.com/tech/697085-comment-exporter-lhistorique-des-discussions-whatsapp.html" \t "_blank)

Je ne peux pas exporter automatiquement toute la conversation depuis cette interface, mais voici **comment procéder manuellement** pour garder une trace complète de ton échange :

**1. Copier-coller manuel**

* **Sélectionne** tout le texte de la conversation dans la fenêtre de chat (Ctrl+A puis Ctrl+C).
* **Colle** le contenu dans un fichier texte (.txt), Word (.docx) ou dans un document Google Docs.
* **Sauvegarde** ce fichier sur ton ordinateur ou dans le cloud.

**2. Impression ou export PDF**

* Après avoir tout sélectionné, tu peux aussi :
  + Coller dans Word ou Google Docs puis faire **Fichier > Exporter en PDF**.
  + Ou utiliser la fonction **Imprimer** de ton navigateur (Ctrl+P) et choisir **Enregistrer au format PDF**.

**3. Extensions ou outils tiers (si disponibles)**

* Certains navigateurs proposent des extensions pour sauvegarder une page web ou une conversation en un clic (ex : "Save as PDF", "Webpage to PDF").
* Si tu utilises une plateforme avec une fonction d’export intégrée (comme ChatGPT officiel ou WhatsApp), suis le menu "Paramètres > Exporter" ou "Exporter la conversation"[1](https://chatgpt-info.fr/exporter-conversations/).

**4. Sauvegarde des fichiers joints**

* Si tu as partagé ou reçu des fichiers (comme des Excel), pense à les télécharger et à les conserver dans le même dossier que ton archive texte.

**Remarque**

* Il n’existe pas de bouton d’export automatique natif sur cette interface.
* Pour une conservation complète et structurée, le copier-coller manuel reste la méthode la plus fiable.

**En résumé :**  
Sélectionne toute la conversation, copie-la, colle-la dans un document texte ou PDF, et sauvegarde-le.  
C’est la méthode universelle, rapide et sûre pour garder une trace de tous tes échanges ici[1](https://chatgpt-info.fr/exporter-conversations/)[2](https://community.openai.com/t/is-there-a-way-i-can-export-every-detail-from-a-full-conversation-thread-to-a-new-one-so-i-can-continue-the-chat/1068326).

Partager

Exporter

Réécrire

Questions liées

Comment exporter toute cette conversation dans un fichier Excel ou PDF

@startuml

' =========================

' ENTITES PRINCIPALES RH

' =========================

class Centre {

+id\_centre : int

+nom\_centre : string

+code\_centre : string

}

class Agent {

+id\_agent : int

+reference : string

+nom : string

+prenom : string

+trigram : string

+date\_naissance : date

+nationalite : string

+actif : bool

}

class Licence {

+id\_licence : int

+num\_licence : string

+type\_licence : string

+date\_delivrance : date

+date\_validite : date

+statut : string

+renouvellement : date

+suspension : date

+retrait : date

}

class Qualification {

+id\_qualif : int

+type\_qualif : string

+date\_obtention : date

+date\_validite : date

+statut : string

}

class Mention {

+id\_mention : int

+type\_mention : string

+date\_obtention : date

+date\_validite : date

+statut : string

}

class CertificatMed {

+id\_certif : int

+date\_visite : date

+validite : date

+classe : string

+restriction : string

+suspension : date

}

class Module {

+id\_module : int

+module\_type : string

+module : string

+item : string

+numero : string

+sujet : string

+date : date

+validite : date

+support : string

+precisions : string

}

class Organisme {

+id\_org : int

+nom\_organisme : string

+type\_organisme : string

+agrement : string

}

class Formation {

+id\_formation : int

+date : date

+annee : int

+duree : float

+resultat : string

}

class Evaluation {

+id\_eval : int

+type\_eval : string

+annee : int

+date : date

+resultat : string

}

class Habilitation {

+id\_hab : int

+type\_hab : string

+date\_obtention : date

+date\_expiration : date

+statut : string

}

class Affectation {

+id\_aff : int

+fonction : string

+date\_debut : date

+date\_fin : date

}

' =========================

' GESTION DOCUMENTAIRE

' =========================

class DocumentType {

+id : int

+nom : string

}

class Document {

+id : int

+titre : string

+description : string

+fichier : file

+reference : string

+date\_creation : date

+date\_mise\_a\_jour : date

+est\_archive : bool

}

class DocumentVersion {

+id : int

+numero\_version : string

+fichier : file

+date\_version : date

+commentaire : string

}

class SignatureCircuit {

+id : int

+ordre : int

+organisme : string

+date\_signature : datetime

+statut : string

+commentaire : string

}

' =========================

' RELATIONS RH

' =========================

Centre "1" -- "0..\*" Agent : centre

Agent "1" -- "0..\*" Licence : licences

Agent "1" -- "0..\*" CertificatMed : certificats\_medicaux

Agent "1" -- "0..\*" Formation : formations

Agent "1" -- "0..\*" Evaluation : evaluations

Agent "1" -- "0..\*" Habilitation : habilitations

Agent "1" -- "0..\*" Affectation : affectations

Licence "1" -- "0..\*" Qualification : qualifications

Licence "1" -- "0..\*" Mention : mentions

Formation "0..\*" -- "1" Module : module

Formation "0..\*" -- "0..1" Organisme : organisme

Affectation "0..\*" -- "1" Centre : centre

' =========================

' RELATIONS DOCUMENTAIRE

' =========================

DocumentType "1" -- "0..\*" Document : type\_document

Document "1" -- "0..\*" DocumentVersion : versions

Document "1" -- "0..\*" SignatureCircuit : circuit\_signatures

Document "0..\*" -- "0..\*" Centre : centres\_visibles

Document "0..1" -- "1" Agent : responsable\_sms

SignatureCircuit "0..\*" -- "0..1" Agent : agent

@enduml