Projet janvier

Objectif :

Table des matières

1. Étape 1 : Python - Génération de pseudonymes à partir d'un fichier

1.1. B) Création d'un script python

1.1.1. Lecture des données

Programme :

```python
# Lecture des données
1 usage
def readData(file):
    """
    :param file: le chemin d'accès du fichier
    :return: Renvoie une liste des données de chaque ligne avec les éléments distincts dans un sous tableau
    """
    # Création d'une liste
    data_list = []

    # Lecture du fichier
    with open(file, "r") as f:
        # Parcours ligne par ligne
        for line in f:
            # Ajout des données dans la liste
            data_list.append(line.strip().split(";"))
    return data_list


# Exécution de la fonction
print(readData("PROJET-PARTAGER/testEntree.txt"))
```

Résultats :

```
Run  pseudo
"C:\Users\benja\OneDrive - Ogec la Joliverie\BTS SIO B - 1ere année\Maths informatique\Chapitre 3\Exercices fonctions\.venv\Scripts\python.exe"
[['Emile', 'ZOLA', '1840'], ['Victor', 'HUGO', '1802'], ['George', 'SAND', '1804'], ['Jane', 'AUSTEN', '1775'], ['Mark', 'TWAIN', '1835']]

Process finished with exit code 0
```

La fonction readData(file) récolte les données du fichier « testEntree.txt » ligne par ligne. Ces données sont segmentées dans un sous tableau de la manière suivante : prénom, nom, date.

1.1.2. Traitement des données et Ecriture du résultats

Programme :

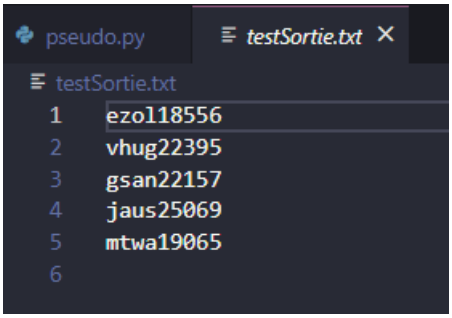
```python
def calculerAge(data_list,i):
    '''
    Cette fonction calcule l'âge de chaque personnalité
    :param data_list: liste des personnalités
    :return: liste des personnalités avec l'âge
    '''
    data_list[i].append(2025 - int(data_list[i][2]))
    return data_list[i][3]

pseudo_list = []
def genererPseudo(data_list):
    '''
    Cette fonction génère un pseudo pour chaque personnalité
    :param data_list: liste des personnalités
    :return: liste des pseudo
    '''
    for i in range(len(data_list)):
        pseudo = ""
        prenom = data_list[i][0][0] #premier caractère du prénom
        nom = data_list[i][1][0:3] #3 premiers caractères du nom
        valeur_aletoire = str(random.randint(10, 99))
        age= calculerAge(data_list,i)
        pseudo = prenom+nom+str(age)+valeur_aletoire
        pseudo = pseudo.lower()#convertir en minuscule
        pseudo_list.append(pseudo)
        print(pseudo)
genererPseudo(data_list)

#-3 Sauvegarde des données
def saveData(pseudo_list):
    '''
    Cette fonction sauvegarde les pseudos dans un fichier texte
    :param pseudo_list: liste des pseudos
    '''
    with open("testSortie.txt", "w") as f:
        for pseudo in pseudo_list:
            f.write(pseudo + "\n")

saveData(pseudo_list)
```

La fonction genererPseudo génère un pseudo pour chaque personnalité en concaténant le premier caractère du prénom, les trois premiers caractères du nom, l'âge et une valeur aléatoire entre 10 et 99. Les pseudos sont ensuite stockés dans une liste pseudo_list et sont par la suite stocker dans un fichier testSortie.txt

Résultat de la sortie :

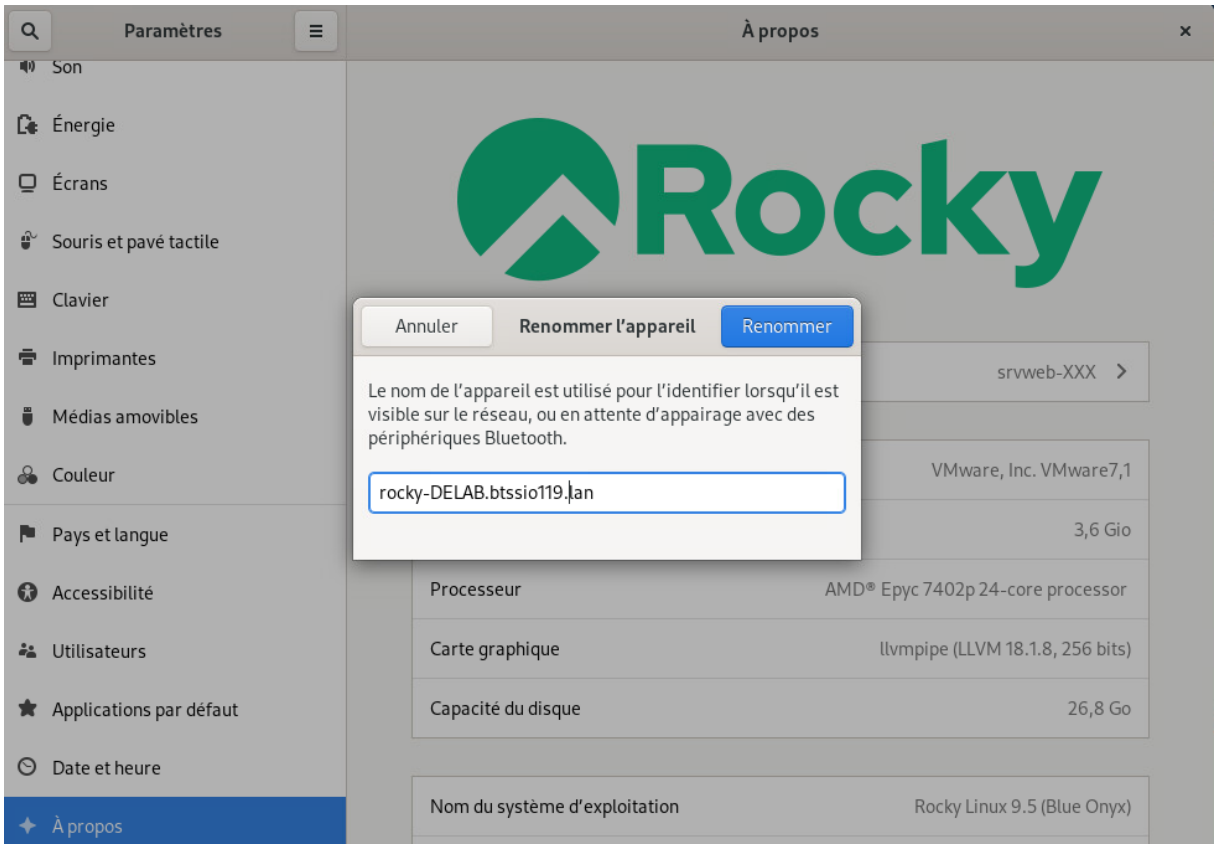
```
PS C:\Users\33698\OneDrive\Bureau\PROJET_Python> & C:/
[['Emile', 'ZOLA', '1840'], ['Victor', 'HUGO', '1802']
ezol18556
vhug22395
gsan22157
jaus25069
mtwa19065
PS C:\Users\33698\OneDrive\Bureau\PROJET_Python>
```

Sauvegardes dans un fichier testSortie.txt :

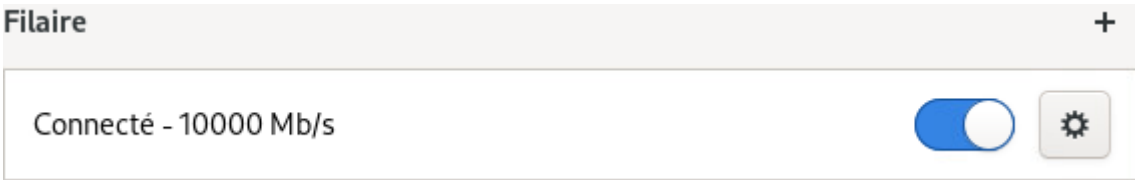


2. Étape 2 : Serveur Mariadb – mise en place de la BDD

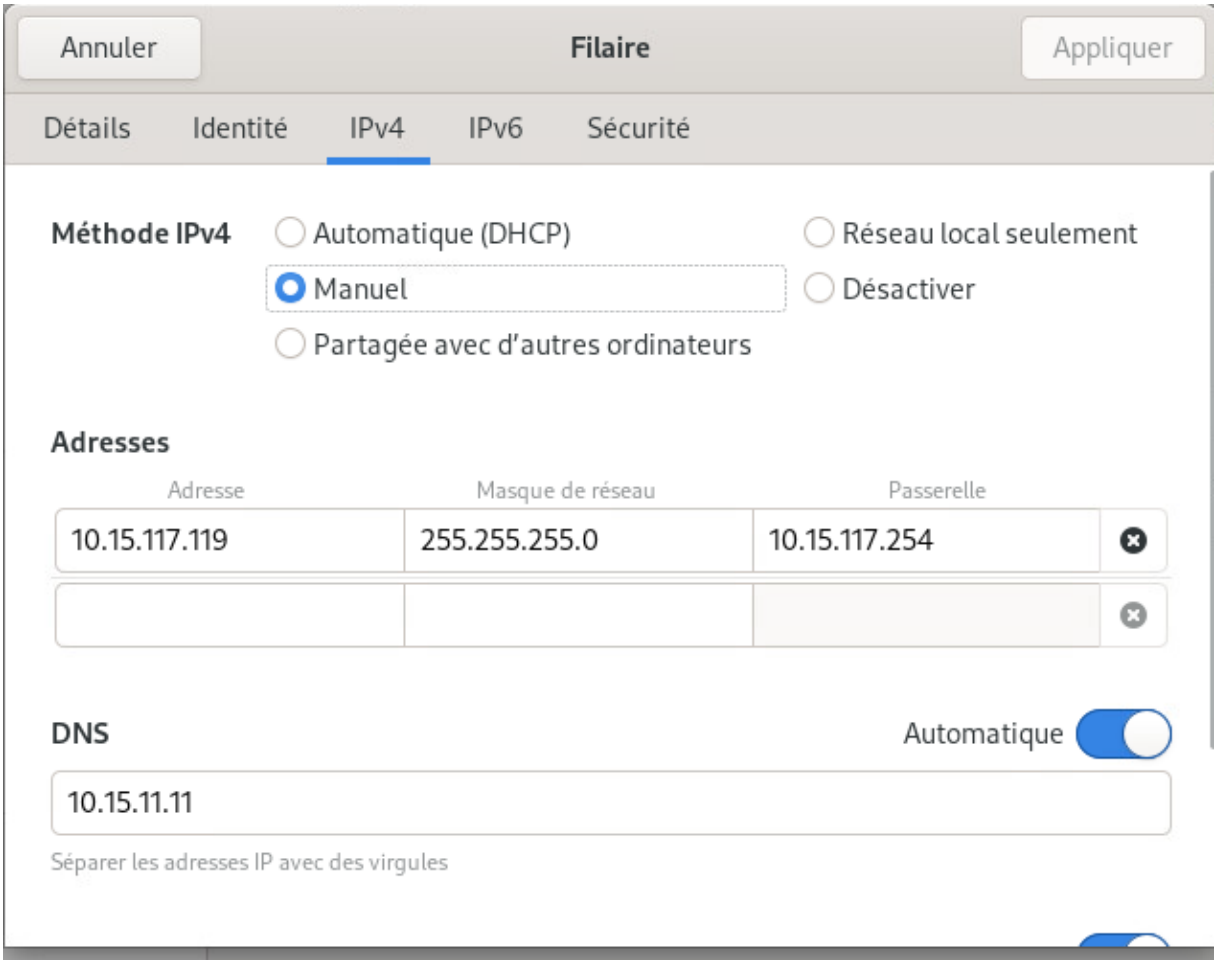
1) Je lance la VM depuis vsphere. Je me connecte au compte btssio puis je vais dans les paramètres de la VM. Je me dirige vers la section « A Propos » dans laquelle je peux modifier le nom de ma machine en :



2) Ensuite, on se rend dans partie réseau située dans « Paramètres ». Puis on clique sur l'engrenage suivant :



Après cela, on clique sur IPv4 et on met la méthode IPv4 en « Manuel ». Dans la section « Adresses », on écrit les données suivantes. A noter que l'adresse réseau de Plot-J-517 (du VLAN) est sur 24 bits. C'est pourquoi les 3 premiers octets sont bloqués.

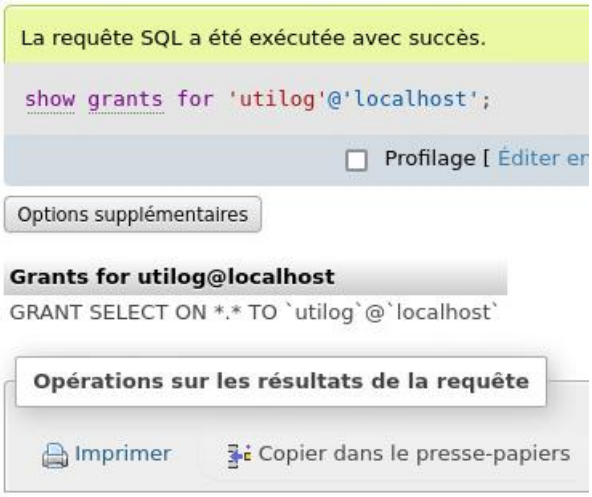


3) Je me connecte à PHPMYADMIN et je créé une base de donnée « logs_web »

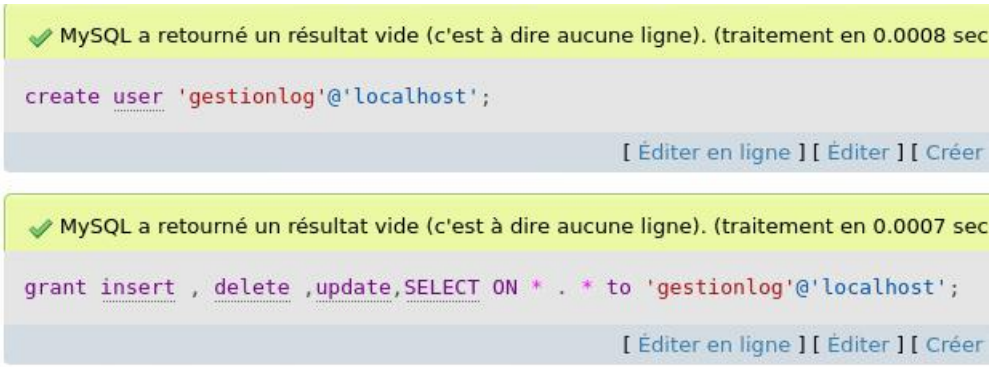
4) Création des utilisateurs en leur attribuant leurs droits respectifs.

a. L'utilisateur « utilog » possède seulement le droit SELECT.

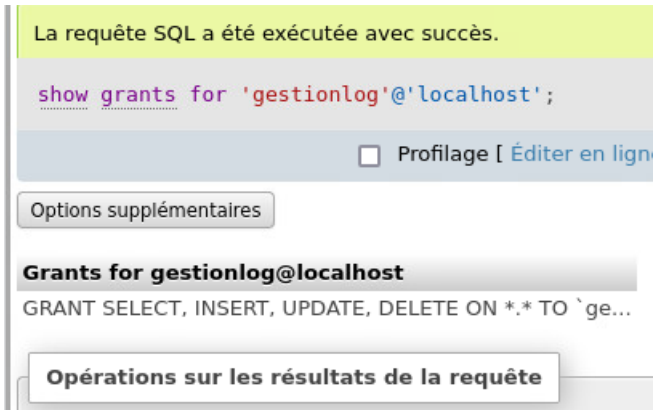
b. Vérification des droits attribués à « utilog ».



c. L'utilisateur « gestionlog » possède quant à lui les droits INSERT, UPDATE, DELETE, et SELECT. Il a donc un accès complet à l'administration de la base de données.



d. Vérification des droits attribués à « gestionlog ».



MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0.0082 seconde(s).)

```
create table employes ( id_employes int primary key not null AUTO_INCREMENT, nom
VARCHAR(50) not null, prenom varchar(50) not null, email varchar(100) UNIQUE,
adresse_ip varchar(15)UNIQUE, date_creation DATETIME DEFAULT NOW(), date_modif
DATETIME ON UPDATE NOW() );
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0.0072 seconde(s).)

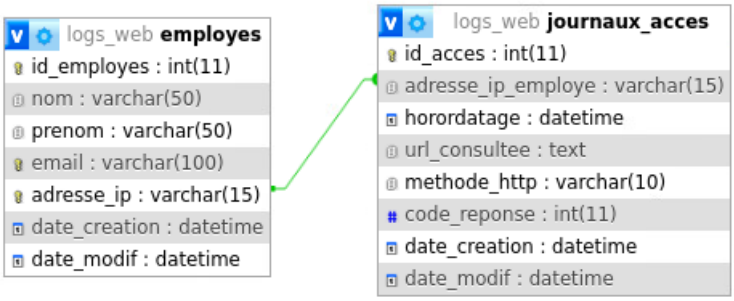
```
create table journaux_acces ( id_acces int primary key not null AUTO_INCREMENT,
adresse_ip_employe varchar(15), horordatage datetime, url_consultee TEXT NOT NULL,
methode_http varchar(10), code_reponse int , date_creation DATETIME DEFAULT NOW(),
date_modif DATETIME ON UPDATE NOW(), FOREIGN KEY (adresse_ip_employe) REFERENCES
employes(adresse_ip) );
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

4 lignes insérées.
Identifiant de la ligne insérée : 4 (traitement en 0.0025 seconde(s).)

```
insert into employes(nom,prenom,email,adresse_ip,date_creation,date_modif)
VALUES('DUFONT','Marie','marie.dufont@example.com','192.168.2.2','2024-01-01 10:00:00','2024-01-01
10:00:00'), ('DESBOIS','Paul','paul.desbois@example.com','192.168.2.1','2024-01-01
10:05:00','2024-01-01 10:05:00'),
('DURANTON','Quentin','quentin.duranton@example.com','192.168.2.3','2024-01-01 10:10:00'
,'2024-01-01 10:10:00'),
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]



Affichage des lignes 0 - 3 (total de 4, traitement en 0.0003 seconde(s).)

```
SELECT * from employes;
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

	id_employes	nom	prenom	email	adresse_ip	date_creation	date_modif
Éditer Copier Supprimer	1	DUFONT	Marie	marie.dufont@example.com	192.168.2.2	2024-01-01 10:00:00	2024-01-01 10:00:00
Éditer Copier Supprimer	2	DESBOIS	Paul	paul.desbois@example.com	192.168.2.1	2024-01-01 10:05:00	2024-01-01 10:05:00
Éditer Copier Supprimer	3	DURANTON	Quentin	quentin.duranton@example.com	192.168.2.3	2024-01-01 10:10:00	2024-01-01 10:10:00
Éditer Copier Supprimer	4	LEJAUNE	Laurence	laurence.lejaune@example.com	192.168.2.100	2024-01-01 10:15:00	2024-01-01 10:15:00

Tout cocher *Avec la sélection :* Éditer Copier Supprimer Exporter

2 lignes insérées.
Identifiant de la ligne insérée : 2 (traitement en 0.0021 seconde(s).)

```
insert into journaux_acces(adresse_ip_employe,horordatage,url_consultee,methode_http , code_reponse,
date_creation,date_modif) values('192.168.2.2','2024-01-02 10:00:00','http://example.com/page1',
'GET',200,'2024-01-02 10:00:00','2024-01-02 10:00:00'), ('192.168.2.1','2024-01-02
10:05:00','http://example.com/page2', 'POST',201,'2024-01-02 10:05:00','2024-01-02 10:05:00');
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

Affichage des lignes 0 - 1 (total de 2, traitement en 0.0002 seconde(s).)

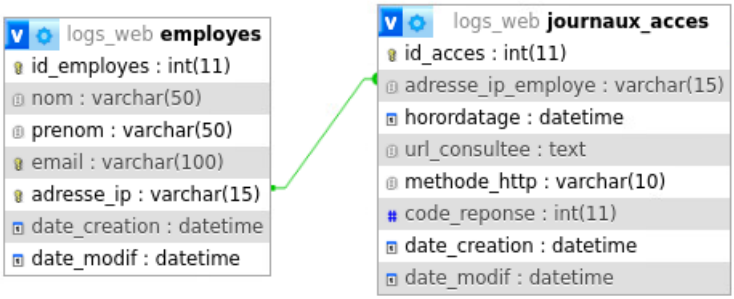
```
select * from journaux_acces;
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

	id_acces	adresse_ip_employe	horordatage	url_consultee	methode_http	code_reponse	date_creation	date_modif
Éditer Copier Supprimer	1	192.168.2.2	2024-01-02 10:00:00	http://example.com/page1	GET	200	2024-01-02 10:00:00	2024-01-02 10:00:00
Éditer Copier Supprimer	2	192.168.2.1	2024-01-02 10:05:00	http://example.com/page2	POST	201	2024-01-02 10:05:00	2024-01-02 10:05:00



```
● mariadb.service - MariaDB 10.5 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: disabled)
     Active: active (running) since Tue 2025-01-07 12:23:33 CET; 1min 42s ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 4100 ExecStartPre=/usr/libexec/mariadb-check-socket (code=exited, status=0/SUCCESS)
    Process: 4123 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir mariadb.service (code=exited>
    Process: 4170 ExecStartPost=/usr/libexec/mariadb-check-upgrade (code=exited, status=0/SUCCE>
   Main PID: 4158 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 8 (limit: 23019)
     Memory: 69.9M
        CPU: 156ms
     CGroup: /system.slice/mariadb.service
             └─4158 /usr/libexec/mariadbd --basedir=/usr

janv. 07 12:23:32 srvweb-XXX systemd[1]: Starting MariaDB 10.5 database server...
janv. 07 12:23:32 srvweb-XXX mariadb-prepare-db-dir[4123]: Database MariaDB is probably initial>
janv. 07 12:23:32 srvweb-XXX mariadb-prepare-db-dir[4123]: If this is not the case, make sure t>
janv. 07 12:23:33 srvweb-XXX systemd[1]: Started MariaDB 10.5 database server.
```

```
<Directory /usr/share/phpMyAdmin/>
   AddDefaultCharset UTF-8
   Require local
   Require ip 10.15.117.119
</Directory>

<Directory /usr/share/phpMyAdmin/setup/>
   Require local
   Require ip 10.15.117.119
```

[Lecture de 49

```
#
# Allow server to accept connections o
#
#bind-address=0.0.0.0
bind-address=10.15.117.119
#
# Optional setting
```

1 ligne insérée.
Identifiant de la ligne insérée : 3 (traitement en 0.0013 seconde(s).)

```
insert into journaux_acces(adresse_ip_employe,
horordatage,url_consultee,methode_http,code_reponse,date_creation,date_modif)
VALUES('192.168.2.3','2024-01-02 10:10:00','http://example.com/page3','GET',202,'2024-01-02
10:10:00','2024-01-02 10:10:00');
```

[Éditer en ligne] [Éditer] [Créer le code source

Affichage des lignes 0 - 2 (total de 3, traitement en 0.0002 seconde(s).)

```
select * from journaux_acces;
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

	id_acces	adresse_ip_employe	horordatage	url_consultee	methode_http	code_reponse	date_creation	date_modif
Éditer Copier Supprimer	1	192.168.2.2	2024-01-02 10:00:00	http://example.com/page1	GET	200	2024-01-02 10:00:00	2024-01-02 10:00:00
Éditer Copier Supprimer	2	192.168.2.1	2024-01-02 10:05:00	http://example.com/page2	POST	201	2024-01-02 10:05:00	2024-01-02 10:05:00
Éditer Copier Supprimer	3	192.168.2.3	2024-01-02 10:10:00	http://example.com/page3	GET	202	2024-01-02 10:10:00	2024-01-02 10:10:00

1 ligne supprimée. (traitement en 0.0018 seconde(s).)

```
DELETE from journaux_acces where id_acces = 3;
```

[Édit

3. Étape 3 : Analyse des fichiers de textes de login

Les fichiers au format `log_proxy_yyyy-mm-dd.txt` contiennent des informations structurées ligne par ligne. Voici les éléments disponibles sur chaque ligne et leur correspondance avec les champs possibles dans une base de données.

Informations présentes dans les logs

1. **Heure (HH:MM:SS)**

a. Extrait : `00:10:58`
b. Description : Heure de la requête effectuée par le client.
c. Champ suggéré : `timestamp` (horodatage).

2. **Adresse IP**
   a. Extrait : `192.168.2.100`
   b. Description : Adresse IP du client effectuant la requête.
   c. Champ suggéré : `client_ip`.
3. **Méthode HTTP**
   a. Extrait : PUT
   b. Description : Méthode utilisée pour la requête (GET, POST, PUT, DELETE, etc.).
   c. Champ suggéré : `http_method`.
4. **Code HTTP**
   a. Extrait : `200`
   b. Description : Code de réponse HTTP indiquant le résultat de la requête (200, 404, 500, etc.).
   c. Champ suggéré : `http_status`.
5. **URL cible**
   a. Extrait : https://www.netflix.com
   b. Description : URL visée par la requête.
   c.
   d. Champ suggéré : `url`.
6. **Taille de la requête/réponse (octets)**
   a. Extrait : `231+2460`
   b. Description : Taille des données envoyées et reçues, séparées par un +

- `Champs suggérés :`
  - `request_size (taille des données envoyées).`
  - `response_size (taille des données reçues).`
- **`Statut de traitement`**
- `Extrait : OK`

- Description : Indication sur le traitement ou la réception réussie de la requête.
- Champ suggéré : processing_status.

Element log	extrait	Champ de la base de données
heure	00:10:58	timestamp
Adresse Ip	192.168.2.100	Client_ip
Methode http	PUT	http_method
Code Http	200	Http_status
Url cible	https://www.netflix.com	url
Taille requête/réponse	231+2460	Request_size Response_size
Statut de traitement	OK	Processing_status

3.1.1. Informations non utilisées dans cette étape

- **Statut de traitement (OK)** : Bien que disponible, ce champ peut ne pas être pertinent pour l'analyse initiale si le code HTTP fournit déjà une indication suffisante sur l'état de la requête (succès, redirection, échec).
- **Combinaison requête/réponse (231+2460)** : Si l'analyse ne nécessite pas de séparer les tailles des données, cet élément pourrait être simplifié en une taille totale.

Justification : Ces informations peuvent être redondantes ou non prioritaires selon les objectifs spécifiques de l'analyse.

4. Etape 4 :

```
SELECT client_ip, url, COUNT(*) AS visit_count
FROM log_table
GROUP BY client_ip, url
ORDER BY client_ip, visit_count DESC
```

- **`COUNT(*)`** : Compte le nombre de visites pour chaque combinaison d'utilisateur (`client_ip`) et URL.
- **`GROUP BY`** : Regroupe les données par utilisateur et URL.
- **`ORDER BY`** : Trie les résultats par utilisateur (`client_ip`) et par nombre de visites décroissant (`visit_count DESC`).

client_ip	url	visit_count
192.168.2.100	https://www.netflix.com	19
192.168.2.1	https://www.fiverr.com	16
192.168.2.3	https://invalid-url-example.org	13

5. Étape 5 : Python Analyses des URL les plus consultées

Tout d'abord, on créé une procédure pour lire le fichier « data_logs ». Ayant rencontré des problèmes lors de l'exécution de la fonction readData(), j'ai préféré recréer une nouvelle procédure dans le fichier « url_par_utilisateur ». Ainsi, on importe la bibliothèque « os » qui va nous servir à retrouver le chemin d'accès du fichier. Ensuite, on lit le dossier et on propose sous la forme d'un menu, le fichier qu'on veut lire. Dans le cas où l'index est supérieur à la longueur de la liste, on redemande une nouvelle fois de rentrer un index. Dans l'autre cas, on compare les index dans la liste du dossier puis on lit le fichier correspondant. Les données sont introduites dans la liste_Data.

```python
import os

# Afficher les logs disponibles sous forme d'un menu
data_logs = []
data_Utilisateurs = {}


1 usage
def readLogs(file, liste_Data):
    """
    :param liste_Data: liste des données logs
    :param file: le chemin d'accès du fichier
    :return: Renvoie une liste des données de chaque ligne avec les éléments distincts dans un sous tableau
    """
    # Lecture du dossier
    liste_Dos = os.listdir(file)

    # Menu
    index = 0
    for fichier in liste_Dos:
        print(f"{index} - {fichier}")
        index += 1

    demande = int(input("Pour analyser un fichier rentrer son numéro d'index correspondant. Le premier index est 0."))

    # Vérification que l'index saisi soit correcte
    if demande > len(liste_Dos) - 1:
        print("Il n'y a aucun index correspond au numéro saisie.")
        print(demande)
    # Si c'est le cas alors
    else:
        for i in range(len(liste_Dos)):
            if i == demande:
                with open(f"{file}/{liste_Dos[demande]}", "r") as f:
                    # Parcours ligne par ligne
                    for line in f:
                        # Ajout des données dans la liste
                        liste_Data.append(line.strip().split(" "))
```

Après ça, on exécute une nouvelle fonction, sortData(). Elle prend en paramètre une liste de donnée (liste_Data). Son but va être de trier les informations pour ressortir l'adresse ip, l'url avec le plus de visites, et le nombre qui y correspond. On établit deux listes : une liste qui contiendra les résultats finaux et une autre pour les adresses ip uniques. De ce fait, on incrémente dans une boucle FOR les adresses ip uniques.

```
40  def sortData(liste_Data):
41      """
42      Compte le nombre d'apparitions de chaque adresse IP dans liste_Data
43      et les ajoute dans liste_Users.
44
45      :param liste_Data: Liste contenant des sous-tableaux avec des adresses IP.
46      :return: None
47      """
48      print("----- SORTDATA ------")
49      # Création de deux listes
50      result = []  # liste finale
51      unique_ips = []  # liste d'ip uniques
52
53      # On ajoute les adresses ip dans la liste unique_ips
54      for element in liste_Data:
55          ip = element[1]
56          if ip not in unique_ips:
57              unique_ips.append(ip)
```

Puis, on compte l'apparition des URL pour chaque adresse ip unique. On recherche l'URL qui a le plus de visites parmi la liste url._count Enfin, on ajoute les données souhaitées dans la liste result à chaque tour de boucle FOR. Avant, d'afficher le résultat voulu.

```
    # Comptage des URL en fonction de l'ip
    for ip in unique_ips:
        url_counts = []  # création d'une liste pour compter le nombre d'apparitions des URL pour chaque ip

        # On parcourt chaque sous tableau de liste_Data
        for element in liste_Data:
            if element[1] == ip:
                url = element[4]

                # On vérifie si l'URL est déjà présente dans la liste
                for entry in url_counts:
                    if entry[0] == url:
                        entry[1] += 1
                        break  # Empêche le bloc else de s'exécuter si l'URL est déjà présente
                else:
                    # Si elle n'existe pas, on l'ajoute
                    url_counts.append([url, 1])

        #  On cherche l'URL qui a le plus de visites pour l'ip actuelle
        maxVisites = 0
        url = ""
        for i in range(len(url_counts)):
            if url_counts[i][1] > maxVisites:
                maxVisites = url_counts[i][1]
                url = url_counts[i][0]

        # On ajoute l'utilisateur, l'url, et le nombre de visites
        result.append([ip, url, maxVisites])

    # Affichage du résultat
    print(result)
    for elt in result:
        print(f"L'URL la plus visitée par {elt[0]} est {elt[1]} nombre de fois {elt[2]}.")


readLogs( file: "Fichiers_logs_proxy/Fichiers_logs_proxy", data_logs)
sortData(data_logs)
```

Run
Shortcut
Edit Key

Résultat :

```
"C:\Users\benja\OneDrive - Ogec la Joliverie\BTS SIO B - 1ere année\Maths informatique\Chapitre 3\Exercices fonctions\.venv\Scripts\python.exe" "C:\Users\benja\OneDrive - Ogec la
0 - log_proxy_2024-11-23.txt
1 - log_proxy_2024-11-24.txt
2 - log_proxy_2024-11-25.txt
Pour analyser un fichier rentrer son numéro d'index correspondant. Le premier index est 0.0
----- SORTDATA ------
[['192.168.2.100', 'https://www.shopify.com', 22], ['192.168.2.3', 'https://www.instagram.com', 14], ['192.168.2.1', 'https://service-unavailable.net', 9], ['192.168.2.2', 'https:
L'URL la plus visitée par 192.168.2.100 est https://www.shopify.com nombre de fois 22.
L'URL la plus visitée par 192.168.2.3 est https://www.instagram.com nombre de fois 14.
L'URL la plus visitée par 192.168.2.1 est https://service-unavailable.net nombre de fois 9.
L'URL la plus visitée par 192.168.2.2 est https://www.shopify.com nombre de fois 7.

Process finished with exit code 0
```

```
Pour analyser un fichier rentrer son numéro d'index correspondant. Le premier index est 0.1
----- SORTDATA ------
L'URL la plus visitée par 192.168.2.2 est https://www.netflix.com nombre de fois 97.
L'URL la plus visitée par 192.168.2.3 est https://nxdomain-test.site nombre de fois 98.
L'URL la plus visitée par 192.168.2.1 est https://nxdomain-test.site nombre de fois 103.
L'URL la plus visitée par 192.168.2.100 est https://www.spotify.com nombre de fois 101.

Process finished with exit code 0
```

```
Pour analyser un fichier rentrer son numéro d'index correspondant. Le premier index est 0.2
----- SORTDATA ------
L'URL la plus visitée par 192.168.2.2 est https://www.google.fr nombre de fois 105.
L'URL la plus visitée par 192.168.2.100 est https://www.imdb.com nombre de fois 110.
L'URL la plus visitée par 192.168.2.3 est https://www.disneyplus.com nombre de fois 99.
L'URL la plus visitée par 192.168.2.1 est https://www.nytimes.com nombre de fois 95.
```

6. Étape 6 : Python CSV

Script python :

logproxy.py > extract_data

```python
# Description: Ce script permet d'extraire les données d'un fichier de log et de les sauvegarder dans un fichier CSV

import csv
import os
data_list = []

saisie_date = input(str("Entrez une date au format YYYY-MM-DD : "))
log_file = None

def extract_data(saisie_date):
    '''
    Cette fonction extrait les données du fichier de log
    :param log_file: fichier de log
    :return: liste des données extraites
    '''
    with open(f"log_dossier/log_proxy_{saisie_date}.txt", "r") as f:#ouvrir le fichier en mode lecture
        for line in f.readlines():#parcourir les lignes du fichier
            data = line.strip().split(" ")#séparer les données
            time = data[0]#extraire l'heure
            ip_address = data[1]#extraire l'adresse IP
            url = data[4]#extraire l'URL
            http_method = data[2]#extraire la méthode HTTP
            response_code = data[6]#extraire le code de réponse
            data_list.append([time, ip_address, url, http_method, response_code])#ajouter les données à la liste
    return data_list

extract_data(saisie_date)

def save_data(data_list):
    '''
    Cette fonction sauvegarde les données extraites dans un fichier CSV
    :param data_list: liste des données extraites
    '''
    #extrait les donnees dans un fichier csv avec comme separateur ;
    #creer moi un dossier pour les csv
    if not os.path.exists("csv_dossier"):
        os.makedirs("csv_dossier")
    with open(f"csv_dossier/log_proxy_{saisie_date}.csv", "w", newline="") as f:#ouvrir le fichier en mode écriture
        writer = csv.writer(f, delimiter=';')#définir le délimiteur
        writer.writerow(["date", "heure", "adresse_ip_employe", "url_consultee", "methode_http", "code_reponse"])#écrire l'en-tête
        for data in data_list:#parcourir les données
            writer.writerow(data)#écrire les données

save_data(data_list)
```

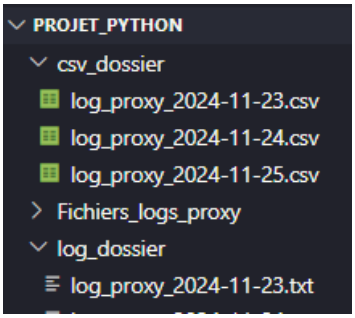
L'utilisateur est invité à saisir une date au format YYYY-MM-DD, les données sont extraites du fichier de log, cette fonction extrait les données du fichier de log, le fichier de log est ouvert en mode lecture.

Les lignes du fichier sont parcourues Les données sont séparées, L'heure, l'adresse IP, l'URL, la méthode HTTP et le code de réponse sont extraits. Les données sont ajoutées à la liste. La liste des données est retournée par la suite les données extraites sont sauvegardées dans un fichier CSV.

Les données extraites sont sauvegardées dans un fichier CSV, le fichier CSV est créé avec comme séparateur ; Un dossier est créé pour les fichiers CSV le fichier est ouvert en mode écriture Le délimiteur est défini L'en-tête est écrit Les données sont parcourues les données sont écrites dans le fichier, le fichier est fermé

Après exécution :

```
PS C:\Users\33698\OneDrive\Bureau\PROJET_Python> & C:/Users/33
Entrez une date au format YYYY-MM-DD : 2024-11-25
PS C:\Users\33698\OneDrive\Bureau\PROJET_Python>
```



csv_dossier > log_proxy_2024-11-25.csv

```
1  date;heure;adresse_ip_employe;url_consultee;methode_http;code_reponse
2  00:00:20;192.168.2.2;https://www.google.fr;PUT;OK
3  00:00:24;192.168.2.2;https://www.office.com;GET;OK
4  00:01:18;192.168.2.100;https://www.disneyplus.com;GET;OK
5  00:01:24;192.168.2.2;https://www.office.com;POST;OK
6  00:01:35;192.168.2.2;https://www.nytimes.com;GET;OK
7  00:03:48;192.168.2.2;https://www.office.com;POST;OK
8  00:03:54;192.168.2.2;https://www.google.fr;POST;OK
9  00:04:17;192.168.2.3;https://www.office.com;GET;OK
10 00:06:11;192.168.2.2;https://www.office.com;POST;OK
11 00:06:35;192.168.2.2;https://www.google.fr;POST;OK
12 00:07:36;192.168.2.1;https://www.nytimes.com;DELETE;OK
13 00:08:25;192.168.2.3;https://www.office.com;GET;OK
```

7. Étape 7 : Python : génération de scripts SQL

Tout d'abord, on importe les bibliothèques « os » et « csv » dans le nouveau fichier « generate_sql_insert.py ». Ensuite, on rédige une fonction « readDos » qui va servir à analyser un fichier, choisi en fonction de son index. Cet index va servir à retourner la date du fichier, contenu dans le nom du fichier.

Description plus précise : On parcourt le dossier à partir de la ligne 13, et on créé une liste qui contiendra les dates de chaque fichier. Ensuite, on va afficher sous la forme d'un menu le fichier à analyser avec son index en préfixe. Puis, on demande quel fichier doit être analyser en donnant l'index correspondant. Enfin, on recherche la date correspondant à l'index précédent en parcourant la liste des dates.

Programme (fonction readDos) :

```python
import os
import csv



1 usage
def readDos(file):
    """
    Affiche la liste des fichiers dans un dossier donné et permet à l'utilisateur de sélectionner un fichier.

    :param file: Chemin d'accès au dossier contenant les fichiers.
    :return: La date extraite du nom du fichier sélectionné.
    """
    # Lecture du contenu du dossier
    liste_Dos = os.listdir(file)
    liste_dates = []  # Liste pour stocker les dates extraites des noms de fichiers

    # Affichage des fichiers disponibles sous forme de menu avec index
    index = 0
    for fichier in liste_Dos:
        print(f"{index} - {fichier}")
        # Extraction de la date du nom du fichier
        liste_dates.append(fichier[10:-4])
        index += 1

    # On demande de saisir l'inde correspondant au fichier à lire
    demande = int(input("Pour analyser un fichier, entrez son numéro d'index correspondant. Le premier index est 0."))

    # Récupére la date du fichier sélectionné en fonction de l'index
    for i in range(len(liste_dates)):
        if i == demande:
            date_fichier = liste_dates[i]
            print(date_fichier)  # Affiche la date sélectionnée
            return date_fichier  # Retourne la date pour l'utiliser dans d'autres fonctions
```

La fonction « insert_sql » génère un fichier SQL contenant la commande « INSERT INTO table VALUES » pour ajouter les informations d'un fichier csv qui contient lui-même les données des logs (étape 6).

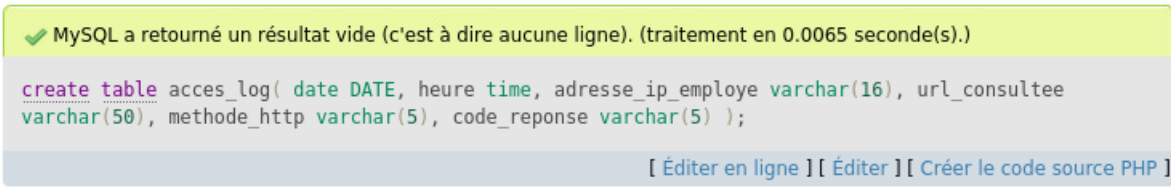
Description plus précise : On récupère la date (la valeur retournée dans la fonction précédente) en paramètre. On commence par vérifier si le dossier « insert_dossier » existe, si ce n'est pas le cas on le créé. Après cela, on ouvre le fichier CSV en ignorant sa première ligne, puis on créé et ouvre un nouveau fichier « log_proxy_{date_fichier}.sql ». À chaque ligne de ce fichier, une commande SQL est inscrite avec les données des lignes CSV correspondant. Enfin, on exécute la programme entier à la ligne 64 avec la fonction « insert_sql » qui prend en paramètre la fonction « readDos » avec comme paramètre le chemin d'accès au dossier.

Programme (fonction insert_sql) :

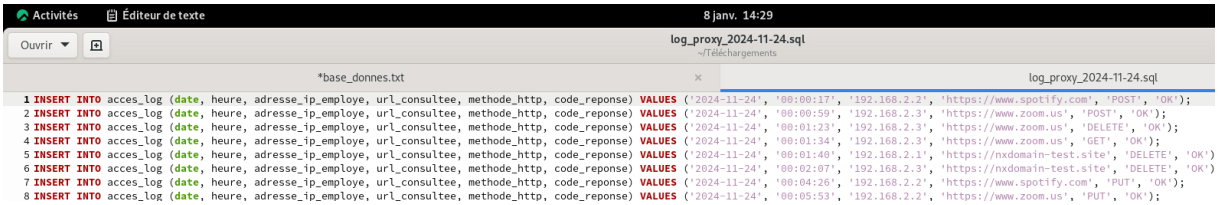
```
35  def insert_sql(date_fichier):
36      """
37      Génère un fichier SQL contenant les commandes d'insertion basées sur un fichier CSV.
38
39      :param date_fichier: Date extraite du nom du fichier log, utilisée pour nommer le fichier SQL.
40      :return: None
41      """
42      # Créé un dossier pour stocker les fichiers SQL si le dossier n'existe pas
43      if not os.path.exists("insert_dossier"):
44          os.makedirs("insert_dossier")
45
46      # Ouvrir le fichier CSV correspondant à la date donnée
47      with open(f"csv_dossier/log_proxy_{date_fichier}.csv", "r") as f:
48          reader = csv.reader(f, delimiter=';')  # Lire le fichier CSV avec un délimiteur ';'
49          next(reader)  # Ignorer la première ligne (en-tête du fichier CSV)
50
51          # Créer et ouvrir le fichier SQL où écrire les commandes d'insertion
52          with open(f"insert_dossier/insert_log_{date_fichier}.sql", "w") as new_f:
53              for row in reader:
54                  # Écrire une commande SQL pour chaque ligne du fichier CSV
55                  new_f.write(
56                      f"INSERT INTO acces_log (date, heure, adresse_ip_employe, url_consultee, methode_http, "
57                      f"code_reponse) VALUES ('{date_fichier}', '{row[0]}', '{row[1]}', '{row[2]}', '{row[3]}', "
58                      f"'{row[4]}');\n"
59                  )
60
61
62  # Exéction de la fonction
63  # Lecture des fichiers disponibles et génération du fichier SQL pour la date sélectionnée
64  insert_sql(readDos("Fichiers_logs_proxy/"))
```

8. Étape 8 : SQL Exécution des scripts:

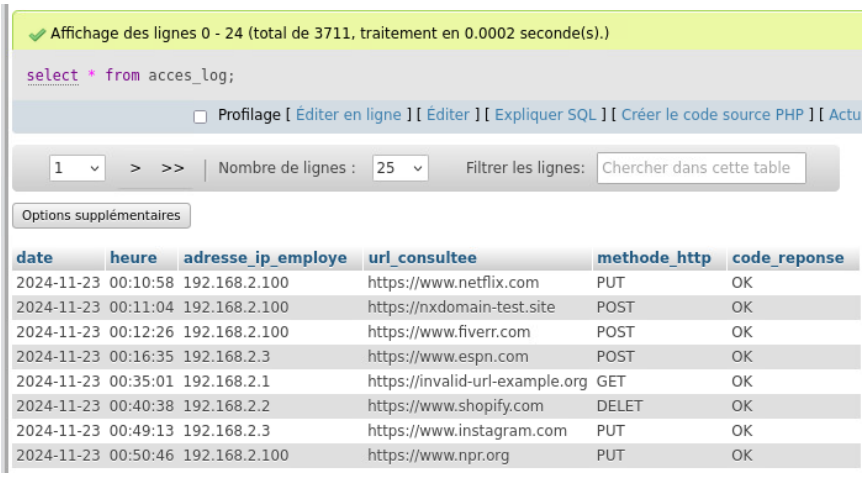
Création de la table acces_ log :



À la suite de la création de la table acces_log on peut ensuite remplir la base de données via la création des scripts sql :



Après remplissage de la base



date	heure	adresse_ip_employe	url_consultee	methode_http	code_reponse
2024-11-23	00:10:58	192.168.2.100	https://www.netflix.com	PUT	OK
2024-11-23	00:11:04	192.168.2.100	https://nxdomain-test.site	POST	OK
2024-11-23	00:12:26	192.168.2.100	https://www.fiverr.com	POST	OK
2024-11-23	00:16:35	192.168.2.3	https://www.espn.com	POST	OK
2024-11-23	00:35:01	192.168.2.1	https://invalid-url-example.org	GET	OK
2024-11-23	00:40:38	192.168.2.2	https://www.shopify.com	DELET	OK
2024-11-23	00:49:13	192.168.2.3	https://www.instagram.com	PUT	OK
2024-11-23	00:50:46	192.168.2.100	https://www.npr.org	PUT	OK

Exécution de la requête faite lors l'étape 4:

Affichage des lignes 0 - 24 (total de 114, traitement en 0.0121 seconde(s).) [adresse_ip_employe: **192.168.2.1.. 192.168.2.100...**]

```
select adresse_ip_employe , url_consultee, COUNT(*) as "visit_count" from acces_log group by
adresse_ip_employe,url_consultee order by adresse_ip_employe, visit_count desc;
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actuali

1 > >> | Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette

Options supplémentaires

adresse_ip_employe 1	url_consultee	visit_count 2
192.168.2.1	https://nxdomain-test.site	103
192.168.2.1	https://www.nytimes.com	95
192.168.2.1	https://www.office.com	87
192.168.2.1	https://www.disneyplus.com	85
192.168.2.1	https://www.spotify.com	85
192.168.2.1	https://www.google.fr	85
192.168.2.1	https://www.salesforce.com	77
192.168.2.1	https://www.imdb.com	76
192.168.2.1	https://www.zoom.us	75
192.168.2.1	https://www.netflix.com	66
192.168.2.1	https://service-unavailable.net	9
192.168.2.1	https://www.fiverr.com	7
192.168.2.1	https://www.garmin.com	6
192.168.2.1	https://invalid-url-example.org	5
192.168.2.1	https://www.reddit.com	4
192.168.2.1	https://www.youtube.com	4
192.168.2.1	https://www.wired.com	3
192.168.2.1	https://www.espn.com	2
192.168.2.1	https://deadlink-example.com	2
192.168.2.1	https://wrongtypedomain.comm	1
192.168.2.1	https://www.outlook.com	1
192.168.2.1	https://www.adobe.com	1
192.168.2.100	https://www.imdb.com	110
192.168.2.100	https://www.google.fr	108
192.168.2.100	https://nxdomain-test.site	101

9. Etape 9 rédaction du modes opératoires :

1.Présentation du format des fichiers de logs :

Les fichiers logs contiennent des informations sous la forme suivante à chaque ligne :
« date;heure;adresse_ip_employe;url_consultee;methode_http;code_reponse ». Le séparateur utilisé entre chaque donnée est un espace. Voici un exemple :

```
00:10:58 192.168.2.100 PUT 200 https://www.netflix.com 231+2460 OK
00:11:04 192.168.2.100 POST 200 https://nxdomain-test.site 503+4939 OK
00:12:26 192.168.2.100 POST 200 https://www.fiverr.com 233+2439 OK
```

2.Utilisation du programme Python :

a) Etapes pour lancer le programme python :

Tout d'abord, on se rend dans le répertoire « dossier_programmes » et on exécute le fichier « logproxy.py » (Étape 6). Pour ce faire, nous pouvons ouvrir le script avec un éditeur de code comme « Visual Studio Code » puis l'exécuter. Ainsi, des fichiers csv vont être généré à partir des fichiers logs au format txt. Lors de l'exécution de ce programme, il faudra rentrer une date au format « YYYY-MM-DD ». Pour séparer ces données, le séparateur « ; » sera utilisé. Un dossier dédié au stockage des fichiers CSV est également généré si nécessaire.

Captures d'écrans :

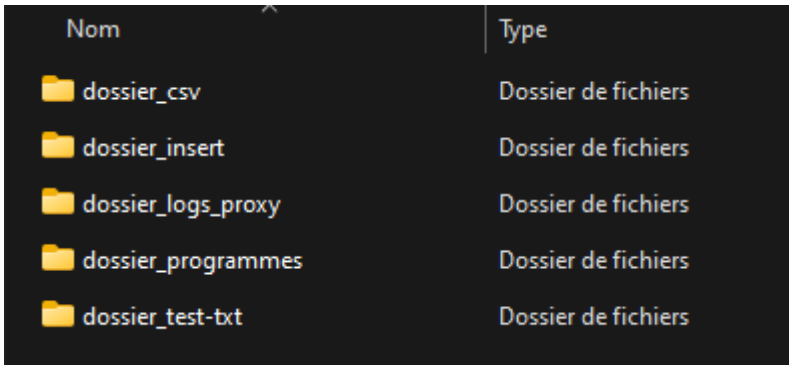


Figure 1 : Contenu du fichier ZIP

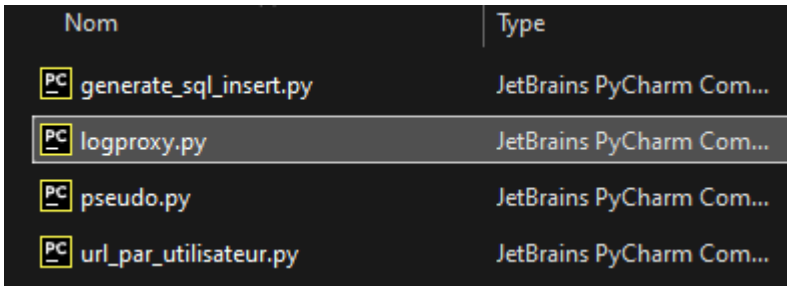


Figure 2 : Contenu du dossier : "dossier_programmes"

b) Explication du menu interactif pour sélection un fichier log

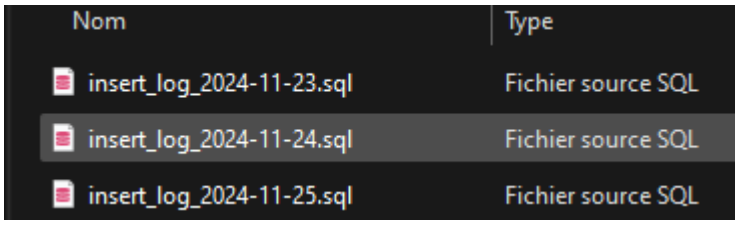
Ensuite, il faut exécuter le programme « generate_sql_insert.py ». Son rôle est de générer un fichier SQL qui ajoutera les informations d'un fichier CSV choisi par l'utilisateur. Lors de l'exécution de ce fichier, un menu est affiché :

```
0 - log_proxy_2024-11-23.csv
1 - log_proxy_2024-11-24.csv
2 - log_proxy_2024-11-25.csv
Pour analyser un fichier, entrez son numéro d'index correspondant. Le premier index est 0.
```

Ici, vous devez renseigner l'index du fichier que vous voulez analyser. A la suite de l'index renseigné (nombre tout à gauche), si vous souhaitez obtenir un fichier SQL contenant les informations du fichier CSV : « logs_proxy_2024-11-24.csv ». Vous devrez taper l'index : « 1 ». La date présente dans le nom du fichier à analyser apparaîtra.

```
Pour analyser un fichier, entrez son numéro d'index correspondant. Le premier index est 0.1
2024-11-24
```

Le fichier SQL apparaîtra dans le dossier : « dossier_insert » qui est créé lors de l'exécution de la fonction s'il n'existe pas encore.



3.Exécution des scripts SQL:

a) Instructions pour exécuter les scripts SQL

Pour se connecter à PhpMyAdmin, vous avez accès aux deux comptes suivants :

- Utilisateur : utilog / mdp : utilog
- Utilisateur : gestionlon / mdp : gestionlog

Vous pourrez voir dans le menu « comptes utilisateurs » vos droits :

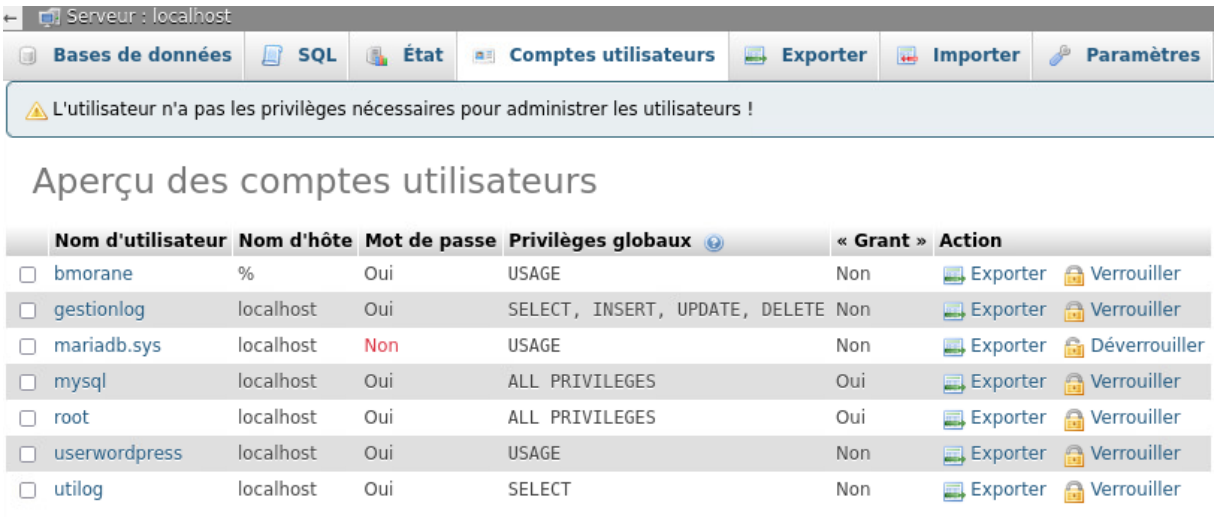


Figure 3 : Droits des utilisateurs

Par la suite, afin d'exécuter les instructions des scripts SQL générés, connectez-vous avec le compte « gestionlog », si ce n'est pas déjà fait. Il faut se rendre dans phpmyadmin, dans la base de données « logs_web » puis dans la table « acces_logs ».

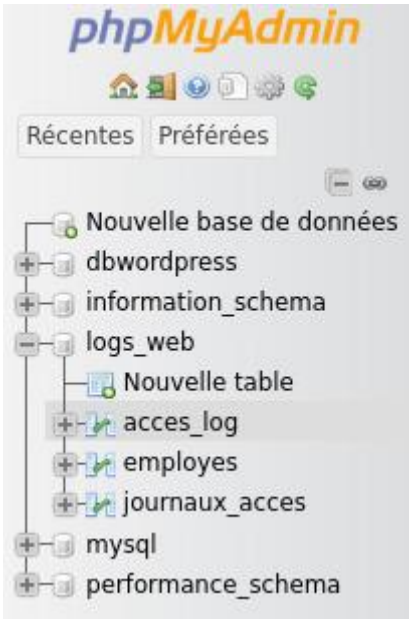


Figure 4 : Localisation de la table "acces_log"

Une fois que vous avez cliqué sur la table « acces_log », il faut cliquer sur le menu « importer », cliquer sur parcourir, « Téléchargements », sélectionner le fichier voulu, et importer le. Pour l'importer définitivement, rendez vous tout en bas de la page et cliquez sur « importer ». Reproduisez cette étape autant de fois qu'il y a de fichiers SQL. C'est-à-dire, encore deux fois.

Figure 5 : Importation du fichier SQL

b) Vérification que les données ont bien été insérée dans la table « acces_logs »

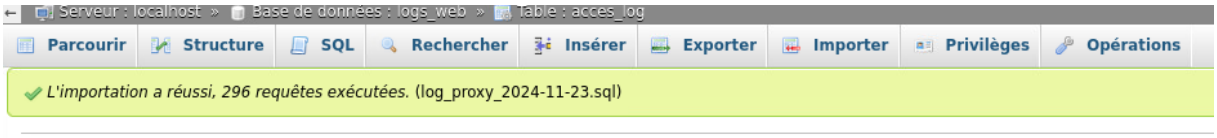
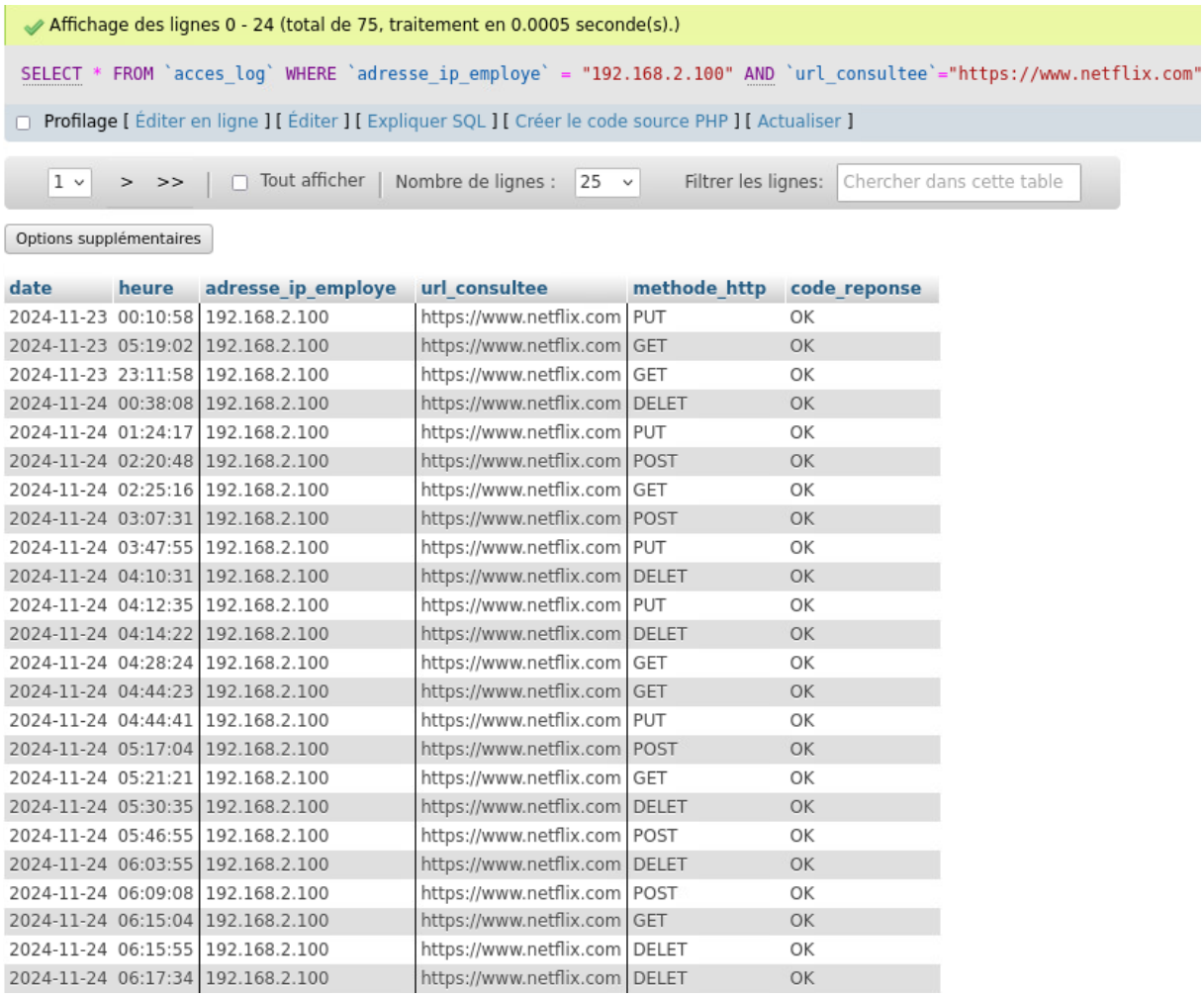


Figure 6 : importation du fichier SQL du 23 novembre 2024

Voir page suivante

4. Test des requêtes sql :

Vous pouvez tester cette requête :



Affichage des lignes 0 - 24 (total de 75, traitement en 0.0005 seconde(s).)

```
SELECT * FROM `acces_log` WHERE `adresse_ip_employe` = "192.168.2.100" AND `url_consultee`="https://www.netflix.com"
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

1 > >> | Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table

Options supplémentaires

date	heure	adresse_ip_employe	url_consultee	methode_http	code_reponse
2024-11-23	00:10:58	192.168.2.100	https://www.netflix.com	PUT	OK
2024-11-23	05:19:02	192.168.2.100	https://www.netflix.com	GET	OK
2024-11-23	23:11:58	192.168.2.100	https://www.netflix.com	GET	OK
2024-11-24	00:38:08	192.168.2.100	https://www.netflix.com	DELET	OK
2024-11-24	01:24:17	192.168.2.100	https://www.netflix.com	PUT	OK
2024-11-24	02:20:48	192.168.2.100	https://www.netflix.com	POST	OK
2024-11-24	02:25:16	192.168.2.100	https://www.netflix.com	GET	OK
2024-11-24	03:07:31	192.168.2.100	https://www.netflix.com	POST	OK
2024-11-24	03:47:55	192.168.2.100	https://www.netflix.com	PUT	OK
2024-11-24	04:10:31	192.168.2.100	https://www.netflix.com	DELET	OK
2024-11-24	04:12:35	192.168.2.100	https://www.netflix.com	PUT	OK
2024-11-24	04:14:22	192.168.2.100	https://www.netflix.com	DELET	OK
2024-11-24	04:28:24	192.168.2.100	https://www.netflix.com	GET	OK
2024-11-24	04:44:23	192.168.2.100	https://www.netflix.com	GET	OK
2024-11-24	04:44:41	192.168.2.100	https://www.netflix.com	PUT	OK
2024-11-24	05:17:04	192.168.2.100	https://www.netflix.com	POST	OK
2024-11-24	05:21:21	192.168.2.100	https://www.netflix.com	GET	OK
2024-11-24	05:30:35	192.168.2.100	https://www.netflix.com	DELET	OK
2024-11-24	05:46:55	192.168.2.100	https://www.netflix.com	POST	OK
2024-11-24	06:03:55	192.168.2.100	https://www.netflix.com	DELET	OK
2024-11-24	06:09:08	192.168.2.100	https://www.netflix.com	POST	OK
2024-11-24	06:15:04	192.168.2.100	https://www.netflix.com	GET	OK
2024-11-24	06:15:55	192.168.2.100	https://www.netflix.com	DELET	OK
2024-11-24	06:17:34	192.168.2.100	https://www.netflix.com	DELET	OK