PROJET DE BASE DE DONNÉES
Software Bug Tracking System

Par : RATSIHOARANA Nomenahitantsoa Amy Andriamalala

PARTIE I : INSTALLER LE PROJET

Ce projet a été conçu avec POSTGRESQL et requiert donc de l'avoir installé

1) Clôner le projet :
   git clone https://github.com/AmysGith/Amy_BD_Projet.git
2) Exécuter creation_tables.sql, puis generationDonneeCSV.ipynb. Pas besoin d'éxécuter le dernier bloc d'import en csv (les fichiers sont déjà compris dans le projet clôné plus tôt)
3) Avant d'insérer les données dans les tables, naviguer vers C:\Program Files\PostgreSQL\16 et y créer un dossier nommé fichierCSV (car c'est ainsi nommé dans insertion_donnees.sql) et y coller tous les fichiers csv contenus dans export_csv
4) Lancer insertion_donnees

Après cela, il est possible de tester le formatage, la suppression des doublons, le pivot,...

PARTIE II :

Spécifications et explications

Utilisation de Faker pour générer les données factices (cohérence des données au maximum, par exemple, générer un nom avec Faker et utiliser ce même nom pour l'email)

Création des données :

- Les projets sont générés en premier comme projets uniques.
- Ensuite, des doublons sont ajoutés en choisissant au hasard parmi ces projets uniques et en copiant exactement leur contenu, à l'exception de l'ID.
- Pour la table projectdeveloper, les couples de clés étrangères sont mélangés aléatoirement car c'est ce couple qui est la clé primaire de la table, et la clé primaire doit être unique

Application des erreurs de format :
Erreurs générales (toutes colonnes)
Sélection des lignes à modifier :

- Pour chaque ligne du DataFrame, la fonction tire un nombre aléatoire entre 0 et 1 (rnd.random())
- Si ce nombre est inférieur à error_rate, la ligne est choisie pour être modifiée
- Cela permet de contrôler la proportion globale de valeurs modifiées.

Choix du type de modification (casse)
Une fois la ligne choisie, la fonction tire un deuxième nombre aléatoire r pour décider comment changer la casse :

- Si $r < 0.33$, la valeur est convertie en majuscules.
- Si $0.33 \leq r < 0.66$, la valeur est convertie en minuscules.
- Si $r \geq 0.66$, la valeur est transformée en casse mixte aléatoire (certaines lettres en majuscules, d'autres en minuscules)

Identification des doublons :

Création du script Python test.py qui permet d'identifier et d'afficher les doublons dans la base, par exemple dans la table project où l'on retrouve 20 couples de doublons

On voit ici une partie de ces couples :

```
C:\Users\NOMENAHITANTSOA\Documents\Amy_BD_Projet\python>python test.py
C:\Users\NOMENAHITANTSOA\Documents\Amy_BD_Projet\python\test.py:14: UserWarning: pandas only supports SQLAlchemy connectable (engine/
connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchem
y.
  df = pd.read_sql_query("SELECT * FROM project", conn)
GROUPE 1:
  Ligne 15 : DEVOLVED DYNAMIC SYNERGY       | RYAN, BURGESS AND PATTERSON             | 2024-02-05 → 2024-08-28 | v3.8.8
  Ligne 41 : DeVolVed DYnaMiC SYneRgy       | ryan, burgess and patterson             | 2024-02-05 → 2024-08-28 | v3.8.8

GROUPE 2:
  Ligne 33 : DEVOLVED INTERACTIVE CONTINGENCY | FORD-BAXTER                           | 2024-07-07 → 2025-06-20 | v1.5.5
  Ligne 60 : DeVOlVeD InTEraCtIvE cOnTINGENCY | ford-baxter                          | 2024-07-07 → 2025-06-20 | v1.5.5

GROUPE 3:
  Ligne 17 : DIVERSE SYSTEMIC INTERFACE     | brown, james and ferrell                | 2024-02-09 → 2025-06-10 | v2.7.8
  Ligne 48 : DIVERSE SYSTEMIC INTERFACE     | brOWN, JaMeS and fERrELL                | 2024-02-09 → 2025-06-10 | v2.7.8

GROUPE 4:
  Ligne 36 : DOWN-SIZED HOMOGENEOUS ABILITY | baker-bowers                           | 2024-11-26 → 2024-12-18 | v4.6.4
  Ligne 55 : dOwn-SIZED hoMOgENeous abILitY | bAkER-BowERs                          | 2024-11-26 → 2024-12-18 | v4.6.4
```

Vérification dans la base de données que ce sont bien des doublons :

36	A Nomenahitantsoa Amy Andriamalala	DOWN-SIZED HOMOGENEOUS ABILITY	baker-bowers	2024-11-26	2024-12-18	4.6.4
55	A Nomenahitantsoa Amy Andriamalala	dOwn-SIZED hoMOgENeous abILitY	bAkER-BowERs	2024-11-26	2024-12-18	4.6.4

TACHE A

FORMATAGE

Avant de pouvoir supprimer efficacement les doublons, il faut corriger les erreurs de format car postgre est sensible à la casse (ProJet n'est pas égal à projet par exemple).
Tous les mots vont commencer par une majuscule et le reste en minuscule, comme ça la comparaison pourra se faire pour la suppression des doublons => utilisation de INITCAP() pour ce faire

Avant :

	projectid [PK] integer	student_fullname text	name text	client text
1	1	RATSIHOARANA Nomenahitantsoa Amy Andriamalala	assimilated user-facing focus group	walker ltd
2	2	RATSIHOARANA Nomenahitantsoa Amy Andriamalala	user-centric needs-based array	HENDERSON, JOHNSON AND ROBINSON
3	3	RATSIHOARANA Nomenahitantsoa Amy Andriamalala	ASSiMilaTED nexT GeNEraTioN CUsTOMeR lOY…	santos, gardner and robinson
4	4	RATSIHOARANA Nomenahitantsoa Amy Andriamalala	adapTIVE welL-mOdulaTeD wORKfORCE	wolfe llc
5	5	RATSIHOARANA Nomenahitantsoa Amy Andriamalala	eXteNDeD wELL-modulAted graPHICAL uSeR in…	DAVIS INC
6	6	RATSIHOARANA Nomenahitantsoa Amy Andriamalala	rObust emPOweRInG WoRKForcE	BLAIR PLC
7	7	RATSIHOARANA Nomenahitantsoa Amy Andriamalala	conFigURAble NeutRaL FraMe	MOntgOMEry, HenSLEY AND RaY

Après :

	projectid [PK] integer	student_fullname text	name text	client text
1	1	Ratsihoarana Nomenahitantsoa Amy Andriamalala	Assimilated User-Facing Focus Group	Walker Ltd
2	2	Ratsihoarana Nomenahitantsoa Amy Andriamalala	User-Centric Needs-Based Array	Henderson, Johnson And Robinson
3	3	Ratsihoarana Nomenahitantsoa Amy Andriamalala	Assimilated Next Generation Customer Loyalty	Santos, Gardner And Robinson
4	4	Ratsihoarana Nomenahitantsoa Amy Andriamalala	Adaptive Well-Modulated Workforce	Wolfe Llc
5	5	Ratsihoarana Nomenahitantsoa Amy Andriamalala	Extended Well-Modulated Graphical User Interface	Davis Inc
6	6	Ratsihoarana Nomenahitantsoa Amy Andriamalala	Robust Empowering Workforce	Blair Plc
7	7	Ratsihoarana Nomenahitantsoa Amy Andriamalala	Configurable Neutral Frame	Montgomery, Hensley And Ray

Suppression des doublons :

1. Commencer par les tables enfants
   - On traite d'abord les tables qui ne sont pas référencées par d'autres tables, comme bugfix et projectdeveloper
   - On choisit les colonnes qui définissent ce qu'est un doublon (tout sauf l'ID).
     Toutes les lignes qui ont exactement les mêmes valeurs pour ces colonnes sont considérées comme un même groupe de doublons.
     On attribue un numéro à chaque ligne du groupe. On utilise ROW_NUMBER() pour numéroter chaque ligne à l'intérieur du groupe. Exemple : dans un groupe de 3 doublons :
     - La première ligne reçoit rn = 1
     - La deuxième ligne reçoit rn = 2
     - La troisième ligne reçoit rn = 3
   - Pour ces tables, on repère les doublons en comparant toutes les colonnes importantes (sauf l'ID ou la clé primaire composée de deux clés étrangères)

   - Dans chaque groupe de doublons, on garde la première ligne (celle avec l'ID le plus petit ou créée en premier) et on supprime les autres

---

2. Traiter les tables parent
   - Les tables parent comme project ou developer sont référencées par d'autres tables
   - On crée une table temporaire pour savoir quel ID doit être gardé et quel ID est un doublon.
   - Cette table sert à mettre à jour toutes les références dans les tables enfants pour que les enfants pointent vers l'ID conservé.

---

3. Supprimer les doublons dans la table parent

Une fois que toutes les références des enfants sont mises à jour, on peut supprimer les doublons dans la table parent sans violer les contraintes de clé étrangère.

---

4. Gestion des tables de liaison (comme projectdeveloper)
   - Après mise à jour des IDs des parents, certains doublons peuvent apparaître dans les tables de liaison
   - On supprime tous les doublons en gardant une seule occurrence par paire (projectid, devid) pour éviter les conflits de clé primaire

---

Correction des emails :

Pour l'instant, il y a des erreurs de format dans les emails :
On génère un nombre aléatoire err compris entre 0 et 1.

On choisit l'action à appliquer en fonction de la valeur de err, en testant les intervalles définis dans les if/elif :

- Si err est entre 0 et 0.1 : on enlève le caractère @
- Si err est entre 0.1 et 0.2 : on enlève le point .
- Si err est entre 0.2 et 0.25 : on ajoute un espace avant @
- Si err est entre 0.25 et 0.3 : on remplace un caractère aléatoire par un chiffre
- Si err est entre 0.3 et 0.35 : on rend l'email null
- Si err est ≥ 0.35 : aucune modification

Illustration dans la base de données :

- Pas de @ :

28	Courtney Velasquez	Courtney Velasquez	courtney.velasquezgmail.com	Psychiatrist

- Pas de . :

33	Wesley White	Wesley White	wesleywhite@gmailcom	Geochemist

- Ajouter un espace avant le @ :

46	William Wilson	William Wilson	william.wilson @gmail.com	Technical Author

- Remplacer par un chiffre :

29	Robert Savage	Robert Savage	robert.savage@8mail.com	Environmental Health Practitioner

- NULL :

49	Katie Anderson	Katie Anderson	[null]	Magazine Journalist

On corrige les mails manquants
Si un email est vide (NULL), on en recrée un automatiquement à partir du nom du développeur

On détecte les emails incorrects
Emails sans @, sans ., avec des espaces ou des caractères invalides. On utilise les regex pour déterminer si un email est valide ou non

On remplace les emails invalides
Quand un email ne respecte pas le format attendu, on le remplace entièrement par un email propre basé sur le nom

Reprise des mêmes exemples après les corrections:

28	Courtney Velasquez	Courtney Velasquez	courtney.velasquez@gmail.com	Psychiatrist
33	Wesley White	Wesley White	wesley.white@gmail.com	Geochemist
46	William Wilson	William Wilson	william.wilson@gmail.com	Technical Author
29	Robert Savage	Robert Savage	robert.savage@gmail.com	Environmental Health Practitioner
49	Katie Anderson	Katie Anderson	katie.anderson@gmail.com	Magazine Journalist

---

CREATION DE L'UDF

Comme le nom de complet de l'étudiant est une colonne présente dans toutes les tables, la création de l'UDF permet de ne pas répéter le même traitement à chaque fois (INITCAP), on appelle juste la fonction fn_normalize_student_name(fullname TEXT)

Avant appel de la fonction :

1	55	56	rATsIhOArANA NOmEnaHiTANtSOa amy anDrIA...
2	5	4	RATSIHOARANA NOMENAHITANTSOA AMY AN...
3	39	54	rAtSiHoARaNA nomENAHITaNTSoA aMY aNdrI...
4	58	42	rATsIhoARAna nomENaHiTANTsoA AMY ANDRI...
5	9	52	ratsihoarana nomenahitantsoa amy andriamalala

Après appel de la fonction :

1	20	28	Ratsihoarana Nomenahitantsoa Amy Andriamalala
2	12	15	Ratsihoarana Nomenahitantsoa Amy Andriamalala
3	5	21	Ratsihoarana Nomenahitantsoa Amy Andriamalala
4	2	35	Ratsihoarana Nomenahitantsoa Amy Andriamalala
5	5	34	Ratsihoarana Nomenahitantsoa Amy Andriamalala

PIVOT

Pour l'instant, dans projectdeveloper, nous n'avons que les IDs des développeurs, et dans project la date de référence de chaque projet. Le pivot va permettre de connaître le nombre de développeurs affectés à chaque projet par année de référence.

Avant :

- Table project

	projectid [PK] integer	name text	startdate date
1	1	Assimilated User-Facing Focus Group	2024-08-17
2	2	User-Centric Needs-Based Array	2024-01-21
3	3	Assimilated Next Generation Customer Loyalty	2024-06-26
4	4	Adaptive Well-Modulated Workforce	2024-07-28
5	5	Extended Well-Modulated Graphical User Interface	2024-03-29

- Table projectdeveloper :

	projectid [PK] integer	devid [PK] integer
1	1	28
2	1	40
3	2	4
4	3	10
5	3	11
6	3	16
7	3	52

Après le pivot :

	project_name text	2024 bigint
1	Adaptive Radical Graphic Interface	1
2	Adaptive Well-Modulated Workforce	1
3	Assimilated Next Generation Customer Loyalty	4
4	Assimilated User-Facing Focus Group	2
5	Configurable Neutral Frame	1
6	Devolved Dynamic Synergy	1

Toutes les dates de référence dans project sont en 2024, c'est la raison pour laquelle il n'y a qu'une colonne 2024.
Par exemple, pour le projet Adaptative Radical Graphic Interface, il a un développeur qui lui est affecté.

TACHE B :
Création de procédures stockées

Avant l'affectation :

```
--Voir au départ si le développeur 12 a déjà été affecté au projet 3
SELECT * FROM projectdeveloper
WHERE projectid = 3 AND devid = 12;
```

	projectid [PK] integer	devid [PK] integer	student_fullname text

=> on voit que ça n'est pas le cas

Après affectation :

```
NOTICE:  Succès : Développeur 12 assigné au projet 3
CALL
```

	projectid [PK] integer	devid [PK] integer	student_fullname text
1	3	12	Ratsihoarana Nomenahitantsoa Amy Andriamalala

Rollback/gestion des erreurs.

```
ERROR:  Association projet 3 / développeur 12 déjà existante
CONTEXT:  fonction PL/pgSQL sp_assign_developer_to_project(integer,integer,text), ligne 24 à RAISE
```

=> On voit ici que le rollback fonctionne : ça relève l'exception disant que ce développeur a déjà été assigné à ce projet.
On peut voir que dans le pivot, le nombre de développeurs assignés au projet 3 a augmenté :
Avant :

3	Assimilated Next Generation Customer Loyalty	4

Après :

3	Assimilated Next Generation Customer Loyalty	5

Vues matérialisées :
=> Voir combien il y a de bugs par projet.
Le calcul repose sur une jointure entre les tables project et bug, suivie d'une agrégation (COUNT). Le résultat est stocké physiquement afin d'éviter de recalculer les jointures et les agrégations à chaque consultation du rapport.
Un index est ajouté sur l'identifiant du projet pour optimiser les accès fréquents.
Visualisation :
Consultation directe de la vue matérialisée affichant le nombre total de bugs pour le projet 1 :

	projectid integer	project_name text	total_bugs bigint
1	1	Assimilated User-Facing Focus Group	1

Vérification du caractère matérialisé :
De nouvelles données sont insérées dans la table bug.
La vue matérialisée n'est pas mise à jour automatiquement, ce qui confirme que les résultats sont stockés physiquement :

	projectid integer	project_name text	total_bugs bigint
1	1	Assimilated User-Facing Focus Group	1

Après exécution d'un rafraîchissement explicite de la vue, les nouvelles données sont alors prises en compte :

	projectid integer	project_name text	total_bugs bigint
1	1	Assimilated User-Facing Focus Group	2

TRIGGERS
Le but est d'empêcher l'insertion ou la mise à jour de bugs avec des valeurs invalides pour severity ou status.

Visualisation :
Faire une insertion où ça ne bloque pas :

```sql
INSERT INTO bug (
    bugid, student_fullname, projectid, title, description, severity, status, createdby
)
VALUES (
    81,'Test User',1,'Bug test valide','Ceci est un test','LOW','OPEN','tester'
);
```

```
INSERT 0 1

Query returned successfully in 155 msec.
```

Faire une insertion invalide (gravité) :

```sql
INSERT INTO bug (
    bugid, student_fullname, projectid, title, description, severity, status, createdby
)
VALUES (82,'Test User',1,'Bug gravité invalide','Test trigger','VERY_HIGH','OPEN','tester'
);
```

```
ERROR:  Gravité invalide : VERY_HIGH
CONTEXT:  fonction PL/pgSQL fn_validate_bug_data(), ligne 4 à RAISE

ERREUR:  Gravité invalide : VERY_HIGH
SQL state: P0001
```

CONCLUSION

Ce projet a permis de travailler sur la création, le nettoyage et la manipulation de données dans un environnement SQL et Python. Nous avons généré des données factices, appliqué et corrigé des erreurs de format, supprimé les doublons et automatisé des traitements avec des UDF, des procédures stockées, des vues matérialisées et des triggers.

Il illustre l’importance de la qualité des données et montre comment des outils programmatiques permettent de gérer efficacement des informations complexes et imparfaites tout en assurant la cohérence et la fiabilité de la base.