Travaux Pratiques PL SQL
Les Triggers

- Travailler avec des déclencheurs dans un contexte métier lié à la gestion de projets.
- Comprendre et appliquer les types de triggers : **BEFORE**, **AFTER**, **INSTEAD OF**.
- Utiliser les déclencheurs pour valider les données, automatiser des actions, et journaliser des opérations.

BEFORE INSERT sur taches

- Créez un déclencheur **BEFORE INSERT** sur la table `taches` qui initialise la colonne `statut` à "Non commencé" si elle est omise.
- **Validation attendue** : Toute insertion sans statut initialise automatiquement `statut` à "Non commencé".

BEFORE UPDATE sur depenses

- Créez un déclencheur **BEFORE UPDATE** sur la table `depenses` pour :
  - Empêcher une mise à jour du montant d'une dépense si le projet associé est marqué comme "Terminé".
  - Retourner une erreur personnalisée via `RAISE_APPLICATION_ERROR` si la règle est violée.
- **Validation attendue** : Les dépenses des projets terminés ne peuvent pas être modifiées.

AFTER INSERT/UPDATE/DELETE sur employes

- Créez un déclencheur **AFTER INSERT OR UPDATE OR DELETE** sur la table `employes` pour journaliser les modifications dans une table `journal_employes` avec les colonnes suivantes :
  - `id_journal` (identifiant auto-incrémenté),
  - `operation` (type d'opération : INSERT, UPDATE, DELETE),
  - `date_operation` (SYSDATE),
  - `details` (texte décrivant les changements : ID et nom de l'employé).
- **Validation attendue** : Toute modification sur `employes` est enregistrée dans `journal_employes`.

BEFORE DELETE sur projets

- Créez un déclencheur **BEFORE DELETE** sur la table `projets` pour :
  - Vérifier qu'aucun employé n'est associé au projet avant sa suppression.
  - Si des employés sont associés, bloquer l'opération avec `RAISE_APPLICATION_ERROR`.
- **Validation attendue** : Un projet ne peut être supprimé que s'il n'a aucun employé associé.

Utilisation de :NEW

- Créez un déclencheur **BEFORE INSERT** sur la table `employes` pour :
  - Initialiser le salaire à 3000 si aucune valeur n'est spécifiée.
  - Ajouter un message de validation dans une table `log_validation`.

Utilisation de :OLD

- Créez un déclencheur **AFTER DELETE** sur la table `taches` pour :
  - Insérer les données supprimées dans une table `historique_taches`.
  - Inclure l'ID de la tâche, la description, et la date limite.

Comparaison entre :NEW et :OLD

- Créez un déclencheur **BEFORE UPDATE** sur la table `depenses` pour :
  - Valider que le montant de la dépense est supérieur à l'ancien montant.
  - Si la condition n'est pas respectée, une erreur doit être levée.

Utilisation de la Clause REFERENCING

- Créez un déclencheur **BEFORE UPDATE** sur la table `depenses` qui :
  - Utilise la clause `REFERENCING` pour renommer `:NEW` et `:OLD` en `nouvelle_valeur` et `ancienne_valeur`.
  - Valide que le montant de la dépense ne dépasse pas 100 000. Si c'est le cas, une erreur doit être levée.

Déclencheur sur Plusieurs Opérations

- Créez un déclencheur **AFTER INSERT OR UPDATE OR DELETE** sur la table `projets` pour journaliser toutes les opérations dans une table `journal_projets`. Les informations incluent :
  - Le type d'opération (INSERT, UPDATE, DELETE).
  - Les valeurs avant et après modification (si applicable).

Déclencheur avec Gestion des Exceptions

- Créez un déclencheur **BEFORE UPDATE** sur la table `taches` qui :
  - Bloque une tentative de changement du statut d'une tâche en "Terminé" si la date limite est dépassée.
  - Si l'opération est bloquée, enregistre l'erreur dans une table `erreurs_taches` avec :
    - L'ID de la tâche.
    - La date de tentative.
    - Un message d'erreur.

Sauvegarde des erreurs

- Créez un déclencheur sur `taches` pour :
  - Bloquer une mise à jour qui changerait le statut en "Terminé" si la date limite est dépassée.
  - En cas de tentative bloquée, insérer une entrée dans une table `erreurs_taches` avec :
    - ID de la tâche,
    - Date de tentative,
    - Message d'erreur.
- **Validation attendue** : Les erreurs sont loguées sans interrompre les autres processus.