|  |  |
| --- | --- |
| **1. Transactions :**Les transactions permettent de regrouper plusieurs opérations SQL en une seule unité de travail.  **START TRANSACTION;**  -- Mettre à jour la moyenne d'un étudiant  **UPDATE** stg SET moy = moy - 3 WHERE mat = 10;  -- Mettre à jour la moyenne d'un autre étudiant  **UPDATE** stg SET moy = moy + 3 WHERE mat = 20;  -- Valider les modifications  **COMMIT;**  -- Si une erreur survient, annuler les modifications  -- **ROLLBACK**; | **2. Gestion des Exceptions :**La gestion des exceptions permet de capturer et de gérer les erreurs SQL,  **DELIMITER $**  **CREATE PROCEDURE** ps\_ajouter\_stg2(**IN** p\_mat INT, **IN** p\_nom CHAR(50), **IN** p\_prenom CHAR(50))  **BEGIN**  -- Gestion des erreurs  **DECLARE EXIT HANDLER FOR** 1048 **SELECT** 'Erreur : valeur nulle non autorisée';  **DECLARE EXIT HANDLER FOR** 1062 **SELECT** 'Erreur : clé dupliquée';  -- Insérer un nouvel étudiant  **INSERT** **INTO** stg (mat, nom, prenom) **VALUES** (p\_mat, p\_nom, p\_prenom);  **SELECT** 'Étudiant ajouté avec succès';  **END$**  **DELIMITER** ;  -- Appel de la procédure  **CALL** ps\_ajouter\_stg2(10, 'ali', 'rachid'); -- Erreur 1062 (clé dupliquée)  **CALL** ps\_ajouter\_stg2(NULL, 'ali', 'rachid'); -- Erreur 1048 (valeur nulle) |
| **3. Curseurs :** Les curseurs permettent de parcourir les résultats d'une requête ligne par ligne  **DELIMITER $**  **CREATE PROCEDURE** somme\_salaire()  **BEGIN**  **DECLARE** finished INT DEFAULT 0;  **DECLARE** v\_salaire INT;  **DECLARE** total INT DEFAULT 0;  -- Déclarer un curseur pour parcourir les salaires  **DECLARE** cur\_salaire **CURSOR** **FOR** SELECT Salaire FROM salaries;  -- Gestionnaire pour la fin du curseur  **DECLARE** **CONTINUE** **HANDLER** **FOR NOT FOUND SET** finished = 1;  -- Ouvrir le curseur  **OPEN** cur\_salaire;  -- Parcourir les résultats  myloop: **LOOP**  **FETCH** cur\_salaire **INTO** v\_salaire;  **IF** finished = 1 **THEN**  **LEAVE** myloop;  **END** IF;  **SET** total = total + v\_salaire;  **END** **LOOP**;  -- Fermer le curseur  **CLOSE** cur\_salaire;  -- Afficher le total  **SELECT** total;  **END**$  **DELIMITER ;**  -- Appel de la procédure  **CALL** somme\_salaire(); | **4. Triggers (Déclencheurs) :** Les triggers sont des procédures automatiques qui s'exécutent en réponse à des événements (INSERT, UPDATE, DELETE) sur une table.  **DELIMITER $**  **CREATE TRIGGER** after\_salaries\_insert  **AFTER** **INSERT** **ON** salaries // **BEFORE** قبل INSERT ON employes  **FOR** **EACH** **ROW**  **BEGIN**  -- Mettre à jour le budget total après chaque insertion  **UPDATE** SalaryBudgets **SET** Total = Total + **NEW**.Salaire **WHERE** num = 1;  **END$**  **DELIMITER** ;  -- Tester le déclencheur  **INSERT** **INTO** salaries **VALUES** (100, 'ali', 'mohamed', '2000-01-01', 'informatique', 5000);  -- |---------------|----------------|---------------|  -- | événement | NEW | OLD |  -- |---------------|----------------|---------------|  -- | INSERT | values | vide |  -- |---------------|---------------- |---------------|  -- | DELETE | vide | values |  -- |---------------|---------------- |---------------|  -- | UPDATE | values aprés | values avant |  -- | | modification | modification  **IF MONTH(NEW.dateReservation) = 7 AND YEAR(NEW.dateReservation) = YEAR(CURDATE()) THEN**  **SIGNAL SQLSTATE '45000'**  **SET MESSAGE\_TEXT = 'Réservations en juillet ne sont pas autorisées.';**  **END IF;**  **SIGNAL SQLSTATE '45000'**  **SET MESSAGE\_TEXT = 'Suppression interdite : dossier de moins de 2 ans** |
| **5. Procédures :**Les procédures stockées peuvent accepter des paramètres en entrée (IN), en sortie (OUT), ou en entrée/sortie (**INOUT**).  **DELIMITER $**  **CREATE** **PROCEDURE** sp\_stg01(**IN** v\_nom VARCHAR(50), **OUT** nombre INT)  **BEGIN**  -- Compter le nombre d'étudiants avec un nom donné  **SELECT** **COUNT(\*) INTO** nombre **FROM** stg **WHERE** nom = v\_nom;  **END$**  **DELIMITER** ;  -- Appel de la procédure  **CALL** sp\_stg01('said', @total);  **SELECT** @total;  La procédure compte le nombre d'étudiants ayant le nom said et stocke le résultat dans la variable @**total** | **6. Fonctions**  **DELIMITER $**  **CREATE** **FUNCTION** mysomme(N INT)  **RETURNS** **INT**  **DETERMINISTIC -- / READS SQL DATA / MODIFIES SQL DATA (**UPDATE**)/ CONTAINS SQL (SELECT CURRENT\_DATE())**  **BEGIN**  **DECLARE** s **INT** **DEFAULT** 0;  **DECLARE** i **INT** **DEFAULT** 1;  **WHILE** (i <= N) **DO**  **SET** s = s + i;  **SET** i = i + 1;  **END** **WHILE**;  **RETURN** s;  **END$**  **DELIMITER ;**  -- Appel de la fonction  **SELECT** mysomme(100); -- Retourne la somme des nombres de 1 à 100 |
| **4. Déclencheur empêchant la suppression de dossiers de moins de 2 ans et non clôturés :**  **DELIMITER $$**  **CREATE TRIGGER before\_delete\_dossier**  **BEFORE DELETE ON Dossier\_terrain**  **FOR EACH ROW**  **BEGIN**  **IF (DATEDIFF(CURDATE(), OLD.date\_ouverture) < 730 AND OLD.date\_cloture IS NULL) THEN**  **SIGNAL SQLSTATE '45000'**  **SET MESSAGE\_TEXT = 'Suppression interdite : dossier de moins de 2 ans et non clôturé.';**  **END IF;**  **END$$**  **DELIMITER ;** | **Requête pour ajouter une contrainte à la table Etablissement**  **ALTER TABLE Etablissement**  **ADD CONSTRAINT chk\_ville**  **CHECK (Ville IN ('Rabat', 'Sale', 'Kenitra'));**  **SELECT n.nom, n.prenom, COUNT(d.iddossier) AS nb\_dossiers**  **FROM Notaire n**  **JOIN Dossier\_terrain d ON n.numn = d.numnotaire**  **GROUP BY n.numn**  **HAVING nb\_dossiers > 200;**  **🔑 En résumé :**  **WHERE** → pour filtrer **les données ligne par ligne** (ex : n.age > 30)  **HAVING** → pour filtrer **des groupes** après les COUNT(), SUM(), etc. |
| **TIMESTAMPDIFF(unit, date1, date2)** | **SELECT nom, DATEDIFF(NOW(), date\_embauche) AS jours\_depuis\_embauche** |