**Fonction pour calculer l'âge d'un employé**

sql

CREATE OR REPLACE FUNCTION calculer\_age(date\_naissance DATE) RETURNS INT AS $$

DECLARE

age INT;

BEGIN

age := EXTRACT(YEAR FROM AGE(current\_date, date\_naissance));

RETURN age;

END;

$$ LANGUAGE plpgsql;

**Déclencheur pour mettre à jour automatiquement le salaire lors d'une promotion**

sql

CREATE OR REPLACE FUNCTION maj\_salaire() RETURNS TRIGGER AS $$

BEGIN

IF NEW.dept\_id <> OLD.dept\_id THEN

UPDATE employe SET salaire = NEW.salaire WHERE nas = OLD.nas;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER maj\_salaire\_trigger AFTER UPDATE ON employe

FOR EACH ROW EXECUTE FUNCTION maj\_salaire();

**Procédure stockée pour afficher les informations d'un employé par son NAS**

sql

CREATE OR REPLACE PROCEDURE afficher\_info\_employe(nas INT) AS $$

DECLARE

employe\_info RECORD;

BEGIN

SELECT \* INTO employe\_info FROM employe WHERE nas = nas;

IF FOUND THEN

RAISE NOTICE 'Nom: %, Prénom: %, Ville: %', employe\_info.nom, employe\_info.prenom, employe\_info.ville;

ELSE

RAISE EXCEPTION 'Employé non trouvé avec le NAS %', nas;

END IF;

END;

$$ LANGUAGE plpgsql;

**Vue pour afficher les employés et leur département associé**

sql

CREATE OR REPLACE VIEW vue\_employe\_departement AS

SELECT e.nas, e.nom, e.prenom, e.sexe, e.date\_naissance, e.taux\_horaire, e.dept\_id, d.nom AS nom\_departement, d.ville AS ville\_departement

FROM employe e

JOIN departement d ON e.dept\_id = d.id;