efm_sgbd2.sql

```
DELIMITER $$
CREATE FUNCTION fn_personnel_rendement(matricule VARCHAR(25), numProj INT)
RETURNS DECIMAL(10,2)
BEGIN
    DECLARE total_heures_travaillees INT;
    DECLARE total_cout_taches DECIMAL(10,2);

    SELECT SUM(tr.nbrHeures), SUM(t.cout)
    INTO total_heures_travaillees, total_cout_taches
    FROM travaille tr
    INNER JOIN tache t ON tr.numTache = t.numTache
    WHERE tr.matricule = matricule AND t.numProj = numProj;

    IF total_cout_taches = 0 THEN
        RETURN NULL;
    ELSE
        RETURN (total_heures_travaillees * 10) / total_cout_taches;
    END IF;
END$$
DELIMITER ;
SELECT fn_personnel_rendement('001', 101) AS rendement_employe_001_projet_101;

-----------------------------------------------------------------------
-- Français
DELIMITER $$
CREATE PROCEDURE ajouter_message_avertissement_fr()
BEGIN
    DECLARE type_operation VARCHAR(50);
    DECLARE current_user VARCHAR(50);
    DECLARE date_operation VARCHAR(50);

    SELECT 'ajout' INTO type_operation; -- Modifier le type d'opération si nécessaire
    SELECT CURRENT_USER() INTO current_user;
    SELECT NOW() INTO date_operation;

    SET @message_fr = CONCAT('Avertissement N° 60000 : Langue française : opération de "',
type_operation, '" bien effectuée par l'utilisateur "', current_user, '" à la date de "',
date_operation, '".');
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = @message_fr;
END$$
DELIMITER ;

-- Anglais
DELIMITER $$
CREATE PROCEDURE add_warning_message_en()
BEGIN
    DECLARE type_operation VARCHAR(50);
    DECLARE current_user VARCHAR(50);
    DECLARE date_operation VARCHAR(50);

    SELECT 'modification' INTO type_operation; -- Modifier le type d'opération si
nécessaire
```

```
    SELECT CURRENT_USER() INTO current_user;
    SELECT NOW() INTO date_operation;

    SET @message_en = CONCAT('Warning No. 60000 : Language English : the "', type_operation, '" operation has been performed by the user "', current_user, '" on the date of "', date_operation, '".');
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = @message_en;
END$$
DELIMITER ;

CALL ajouter_message_avertissement_fr();
CALL add_warning_message_en();
-------------------------------------------------------------------------
DELIMITER $$

CREATE PROCEDURE ps_Projet_supprimer(
    IN p_numProj INT
)
BEGIN
    DECLARE exit_handler BOOLEAN DEFAULT FALSE;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET exit_handler = TRUE;

    START TRANSACTION;

    DELETE FROM travaille WHERE numTache IN (SELECT numTache FROM tache WHERE numProj = p_numProj);
    DELETE FROM tache WHERE numProj = p_numProj;
    DELETE FROM projet WHERE numProj = p_numProj;

    IF NOT exit_handler THEN
        COMMIT;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Suppression du projet et des tâches associées réussie.';
    ELSE
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = CONCAT('Erreur lors de la suppression du projet ', p_numProj, '.');
    END IF;
END$$
DELIMITER ;
-------------------------------------------------------------------------
DELIMITER $$

CREATE PROCEDURE ps_Tache_ajouter(
    IN p_numProj INT,
    IN p_nomTache VARCHAR(25),
    IN p_duree INT,
    IN p_cout DECIMAL(10, 2)
)
BEGIN
    DECLARE numTache INT;
    DECLARE date_debut DATE;

    -- Vérifier si le numéro de projet existe
    SELECT MAX(numTache) + 1 INTO numTache FROM tache;
```

```
102
103     IF p_cout IS NULL THEN
104         SET p_cout = 0.0;
105     END IF;
106
107     SELECT
108         CASE WHEN MAX(dateFin) IS NULL THEN CURDATE()
109              ELSE DATE_ADD(MAX(dateFin), INTERVAL 1 DAY)
110         END AS date_debut
111     INTO date_debut
112     FROM tache
113     WHERE numProj = p_numProj;
114
115     SET @date_fin = DATE_ADD(date_debut, INTERVAL p_duree DAY);
116
117     INSERT INTO tache (numTache, nomTache, dateDeb, dateFin, cout, numProj)
118     VALUES (numTache, p_nomTache, date_debut, @date_fin, p_cout, p_numProj);
119
120     IF ROW_COUNT() > 0 THEN
121         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = CONCAT('Ajout de la tâche réussi.
Numéro de tâche : ', numTache);
122     ELSE
123         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Erreur lors de l''ajout de la tâche.';
124     END IF;
125 END$$
126 DELIMITER ;
127 ----------------------------------------------------------------------
128 DELIMITER $$
129
130 CREATE PROCEDURE ps_Personnel_augmenter(
131     IN p_numProj INT
132 )
133 BEGIN
134     DECLARE exit_handler BOOLEAN DEFAULT FALSE;
135     DECLARE total_augmentation DECIMAL(10, 2) DEFAULT 0.0;
136
137     DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET exit_handler = TRUE;
138
139     START TRANSACTION;
140
141     UPDATE employe e
142     JOIN (
143         SELECT tr.matricule,
144                SUM(tr.nbrHeures) AS total_heures,
145                SUM(t.cout) AS total_cout
146         FROM travaille tr
147         INNER JOIN tache t ON tr.numTache = t.numTache
148         WHERE t.numProj = p_numProj
149         GROUP BY tr.matricule
150         ORDER BY (SUM(tr.nbrHeures) * 10 / SUM(t.cout)) DESC
151         LIMIT 3
152     ) top_employes ON e.matricule = top_employes.matricule
153     SET e.salaire = CASE
154         WHEN e.matricule = top_employes.matricule THEN
155             CASE
```

```
156                 WHEN FIND_IN_SET(e.matricule, (SELECT GROUP_CONCAT(matricule ORDER BY
(SUM(tr.nbrHeures) * 10 / SUM(t.cout)) DESC) FROM travaille tr INNER JOIN tache t ON
tr.numTache = t.numTache WHERE t.numProj = p_numProj)) = 1 THEN
157                     e.salaire * 1.02
158                 WHEN FIND_IN_SET(e.matricule, (SELECT GROUP_CONCAT(matricule ORDER BY
(SUM(tr.nbrHeures) * 10 / SUM(t.cout)) DESC) FROM travaille tr INNER JOIN tache t ON
tr.numTache = t.numTache WHERE t.numProj = p_numProj)) = 2 THEN
159                     e.salaire * 1.01
160                 WHEN FIND_IN_SET(e.matricule, (SELECT GROUP_CONCAT(matricule ORDER BY
(SUM(tr.nbrHeures) * 10 / SUM(t.cout)) DESC) FROM travaille tr INNER JOIN tache t ON
tr.numTache = t.numTache WHERE t.numProj = p_numProj)) = 3 THEN
161                     e.salaire * 1.005
162                 ELSE
163                     e.salaire
164             END
165         ELSE
166             e.salaire
167     END;
168
169     IF NOT exit_handler THEN
170         COMMIT;
171         SELECT SUM(e.salaire * 0.02 + e.salaire * 0.01 + e.salaire * 0.005) INTO
total_augmentation FROM employe e WHERE e.matricule IN (SELECT matricule FROM travaille
WHERE numTache IN (SELECT numTache FROM tache WHERE numProj = p_numProj));
172         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = CONCAT('Augmentation des salaires
effectuée avec succès. Montant total d''augmentation : ', total_augmentation);
173     ELSE
174         ROLLBACK;
175         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = CONCAT('Erreur lors de l''augmentation
des salaires pour le projet ', p_numProj, '.');
176     END IF;
177 END$$
178 DELIMITER ;
179 ------------------------------------------------------------------------
180 CREATE TABLE SalaireLog (
181     Num_auto INT AUTO_INCREMENT PRIMARY KEY,
182     matricule VARCHAR(25),
183     date_modification TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
184     ancien_salaire DECIMAL(10, 2),
185     nouveau_salaire DECIMAL(10, 2),
186     taux DECIMAL(10, 2),
187     utilisateur VARCHAR(50)
188 );
189 ------------------------------------------------------------------------
190 DELIMITER $$
191
192 CREATE TRIGGER tg_salaire_log
193 BEFORE UPDATE ON employe
194 FOR EACH ROW
195 BEGIN
196     DECLARE ancien_salaire DECIMAL(10, 2);
197     SET ancien_salaire = OLD.salaire;
198
199     IF NEW.salaire <> OLD.salaire THEN
200         INSERT INTO SalaireLog (matricule, ancien_salaire, nouveau_salaire, taux,
utilisateur)
201         VALUES (NEW.matricule, ancien_salaire, NEW.salaire, (NEW.salaire - ancien_salaire)
/ ancien_salaire, CURRENT_USER());
```

```
    END IF;
END$$

DELIMITER ;
-----------------------------------------------------------------------
DELIMITER $$

CREATE TRIGGER tg_tache_ajouter
BEFORE INSERT ON tache
FOR EACH ROW
BEGIN
    DECLARE nb_taches INT;
    DECLARE nb_limite_taches INT;

    SELECT COUNT(*) INTO nb_taches FROM tache WHERE numProj = NEW.numProj;
    SELECT nbrLimiteTaches INTO nb_limite_taches FROM projet WHERE numProj = NEW.numProj;

    IF nb_taches >= nb_limite_taches THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Le nombre de tâches dépasse la limite pour ce projet.';
    END IF;
END$$

DELIMITER ;
-----------------------------------------------------------------------
DELIMITER $$

CREATE TRIGGER tg_projet_supprimer
AFTER DELETE ON projet
FOR EACH ROW
BEGIN
    DELETE FROM travaille WHERE numTache IN (SELECT numTache FROM tache WHERE numProj = OLD.numProj);
    DELETE FROM tache WHERE numProj = OLD.numProj;
END$$
DELIMITER ;
-----------------------------------------------------------------------
DELIMITER $$

CREATE TRIGGER tg_projet_ajouter
AFTER INSERT ON projet
FOR EACH ROW
BEGIN
    DECLARE i INT DEFAULT 1;
    DECLARE nbr_taches INT;

    SELECT nbrLimiteTaches INTO nbr_taches FROM projet WHERE numProj = NEW.numProj;

    WHILE i <= nbr_taches DO
        INSERT INTO tache (nomTache, dateDeb, dateFin, cout, numProj)
        VALUES (CONCAT('tache ', i), CURDATE(), DATE_ADD(CURDATE(), INTERVAL 40 DAY), NULL, NEW.numProj);
        SET i = i + 1;
    END WHILE;
END$$
DELIMITER ;
```

```
255 ------------------------------------------------------------------------
256
```