GROUP : 1.9
ID : 201501019 : RAGHAV
201501041 : VIDISH
201501046 : YASH

DATABASE : COMPETITIVE_PROGRAMMING_PLATFORM

FUNCTIONS AND TRIGGERS

--FUNCTION_4: RETURN_TITLE : RETURNS TITLE FROM RATING

```
CREATE OR REPLACE FUNCTION RETURN_TITLE(RATING INTEGER) RETURNS
VARCHAR(20)
AS $BODY$
	DECLARE
		R VARCHAR(20);
	BEGIN
		IF(RATING < 0) THEN
			RETURN 'NOT DEFINE';
		END IF;
		SELECT TITLE INTO R FROM STATUS_TITLE WHERE RATING >=
START_RATING_RANGE AND RATING <=END_RATING_RANGE;
		RETURN R;
	END
$BODY$ LANGUAGE PLPGSQL;
```

--FUNCTION_5: UPDATE_CONTEST_DATETIME : RETURNS TRUE IF DATE AND TIME UPDATED ELSE FALSE

```
CREATE OR REPLACE FUNCTION UPDATE_CONTEST_DATETIME(CID INTEGER,
S_DATE DATE, S_TIME TIME(6) WITHOUT TIME ZONE, E_DATE DATE, E_TIME TIME(6)
WITHOUT TIME ZONE) RETURNS BOOLEAN
AS $BODY$
	DECLARE
		RES INTEGER;
	BEGIN
		IF (NOT EXISTS(SELECT DISTINCT CONTEST_ID FROM CONTESTS
WHERE CONTESTS.CONTEST_ID=CID)) THEN
			RAISE NOTICE 'THAT CONTEST_ID % DOES NOT EXIST' , CID;
			RETURN FALSE;
		ELSE
```

```
                        SELECT
CHECK_VALID_DATETIME(S_DATE,S_TIME,E_DATE,E_TIME) INTO RES;
                        IF(RES>=15) THEN
                                UPDATE CONTESTS SET
START_DATE=S_DATE,START_TIME=S_TIME,END_DATE=E_DATE,END_TIME=E_TIM
E WHERE CONTEST_ID=CID;
                                                RETURN TRUE;
                        ELSE
                                RAISE NOTICE 'START DATE OR TIME IS GREATER THAN
END DATE OR TIME';
                                                RETURN FALSE;
                        END IF;
                END IF;
        END
$BODY$ LANGUAGE PLPGSQL;
```

--FUNCTION_6: UPDATE_PRACTICE_SET_STATUS : RETURNS TRUE IF STATUS UPDATED ELSE FALSE

```
CREATE OR REPLACE FUNCTION UPDATE_PRACTICE_SET_STATUS(CID INTEGER)
RETURNS BOOLEAN
AS $BODY$
        BEGIN
                IF(CID <= 0) THEN
                        RAISE NOTICE 'INVALID CONTEST_ID';
                        RETURN FALSE;
                ELSIF (NOT EXISTS(SELECT DISTINCT CONTEST_ID FROM CONTESTS
WHERE CONTESTS.CONTEST_ID=CID)) THEN
                        RAISE NOTICE 'THAT CONTEST_ID % DOES NOT EXIEST' , CID;
                        RETURN FALSE;
                ELSIF (NOT EXISTS(SELECT DISTINCT CONTEST_ID FROM QUESTIONS
WHERE QUESTIONS.CONTEST_ID=CID)) THEN
                        RAISE NOTICE 'THAT CONTEST_ID % HAS NO QUESTIONS' , CID
;
                        RETURN FALSE;
                ELSE
                        UPDATE QUESTIONS SET PRACTICE_SET_STATUS='1' WHERE
CONTEST_ID=CID;
                        RETURN TRUE;
                END IF;
        END
$BODY$ LANGUAGE PLPGSQL;
```

--FUNCTION_7: CREATE_LEADERBOARD : RETURNS SET OF LEADERBOARD OF GIVEN CONTEST_ID

```
CREATE OR REPLACE FUNCTION CREATE_LEADERBOARD(CID INTEGER) RETURNS
SETOF LEADERBOARD

AS $BODY$

	DECLARE

		R RECORD;

		L LEADERBOARD%ROWTYPE;

		C CONTESTS%ROWTYPE;

		CE_DATE DATE;

		CE_TIME TIME(6) WITHOUT TIME ZONE;

		COUNTER INTEGER := 0;

		COUNTER2 INTEGER := 1;

		LAST_SCORE DECIMAL(5,2) := -1;

	BEGIN

		IF(NOT EXISTS(SELECT CONTEST_ID FROM CONTESTS WHERE
CONTEST_ID=CID))THEN

		RAISE NOTICE ' CONTEST NOT EXISTS';

		RETURN;

		END IF;


		SELECT * INTO C FROM CONTESTS WHERE CONTEST_ID=CID;

		CE_DATE := C.END_DATE;

		CE_TIME := C.END_TIME;


		CREATE TEMPORARY TABLE TMP_LB (HANDLE
VARCHAR(20),CONTEST_ID INTEGER,QUESTION_ID CHAR(1), SCORE DECIMAL(5,2))
ON COMMIT DROP;
```

```
        CREATE TEMPORARY TABLE CREATORS (HANDLE VARCHAR(20)) ON
COMMIT DROP;

        CREATE TEMPORARY TABLE REGISTERS (HANDLE VARCHAR(20)) ON
COMMIT DROP;

        CREATE TEMPORARY TABLE DISQUALIFIED (HANDLE VARCHAR(20))
ON COMMIT DROP;


        FOR R IN (SELECT DISTINCT CREATOR_ID FROM QUESTIONS WHERE
CONTEST_ID=CID)

        LOOP

                INSERT INTO CREATORS (HANDLE) SELECT R.CREATOR_ID;

                RAISE NOTICE 'CREATOR = %',R.CREATOR_ID;

        END LOOP;


        FOR R IN (SELECT * FROM REGISTERS_IN WHERE CONTEST_ID=CID)

        LOOP

                INSERT INTO REGISTERS (HANDLE) SELECT R.HANDLE;

                RAISE NOTICE 'REGISTER = %',R.HANDLE;

        END LOOP;


        FOR R IN (SELECT * FROM REGISTERS_IN WHERE CONTEST_ID=CID
AND DISQUALIFIED_FLAG='1')

        LOOP

                INSERT INTO DISQUALIFIED (HANDLE) SELECT R.HANDLE;

                RAISE NOTICE 'DISQUALIFIED = %',R.HANDLE;

        END LOOP;


        FOR R IN (SELECT
SUBMISSION_ID,HANDLE,CONTEST_ID,QUESTION_ID,SCORE FROM SUBMISSIONS
```

```
WHERE CONTEST_ID=CID AND HANDLE NOT IN(SELECT HANDLE FROM CREATORS)
AND
CHECK_VALID_DATETIME(SUBMISSION_DATE,SUBMISSION_TIME,CE_DATE,CE_TIM
E) >= 0 AND SUBMISSIONS.HANDLE IN (SELECT HANDLE FROM REGISTERS) AND
SUBMISSIONS.HANDLE NOT IN (SELECT HANDLE FROM DISQUALIFIED))

        LOOP

            IF(R.SCORE >= ALL (SELECT SCORE FROM SUBMISSIONS
WHERE HANDLE=R.HANDLE AND CONTEST_ID=R.CONTEST_ID AND
QUESTION_ID=R.QUESTION_ID))THEN

                IF(NOT EXISTS(SELECT SCORE FROM TMP_LB WHERE
HANDLE=R.HANDLE AND CONTEST_ID=R.CONTEST_ID AND
QUESTION_ID=R.QUESTION_ID AND SCORE=R.SCORE))THEN

                    INSERT INTO TMP_LB
(HANDLE,CONTEST_ID,QUESTION_ID,SCORE) SELECT
R.HANDLE,R.CONTEST_ID,R.QUESTION_ID,R.SCORE;

                    RAISE NOTICE 'SUB_ID = %' , R.SUBMISSION_ID;

                END IF;

            END IF;

        END LOOP;



        FOR R IN (SELECT * FROM (SELECT
HANDLE,CONTEST_ID,SUM(SCORE) AS SCORE FROM TMP_LB GROUP BY
HANDLE,CONTEST_ID) AS TMP ORDER BY SCORE DESC)

        LOOP

            IF(R.SCORE <> LAST_SCORE) THEN

                LAST_SCORE := R.SCORE;

                COUNTER := COUNTER2;

            END IF;

            COUNTER2 := COUNTER2 + 1;



            L.HANDLE := R.HANDLE;

            L.CONTEST_ID := R.CONTEST_ID;
```

```
                L.SCORE := R.SCORE;

                L.STANDING := COUNTER;

                RAISE NOTICE 'HANDLE % , SCORE %',R.HANDLE,R.SCORE;


                IF(NOT EXISTS(SELECT * FROM LEADERBOARD WHERE
HANDLE=R.HANDLE AND CONTEST_ID=R.CONTEST_ID)) THEN

                        INSERT INTO LEADERBOARD
VALUES(R.HANDLE,R.CONTEST_ID,R.SCORE,COUNTER);

                        RAISE NOTICE 'INSERT';

                ELSE

                        UPDATE LEADERBOARD SET
STANDING=COUNTER,SCORE=R.SCORE  WHERE HANDLE=R.HANDLE AND
CONTEST_ID=R.CONTEST_ID;

                        RAISE NOTICE 'UPDATE';

                END IF;


                RETURN NEXT L;

        END LOOP;

        RETURN;

    END

$BODY$ LANGUAGE PLPGSQL;
```

--FUNCTION_8: GIVE_MEDALS : RETURNS SET OF HAS_MEDALS OF GIVEN CONTEST_ID

```
CREATE OR REPLACE FUNCTION GIVE_MEDALS(CID INTEGER) RETURNS SETOF
HAS_MEDALS
AS $BODY$
    DECLARE
        R RECORD;
        FLAG INTEGER := 0;
```

```
		LAST_STANDING INTEGER := 0;
		H_M HAS_MEDALS;
	BEGIN
		IF(NOT EXISTS(SELECT CONTEST_ID FROM CONTESTS WHERE
CONTEST_ID=CID))THEN
		RAISE NOTICE ' CONTEST NOT EXISTS';
		RETURN;
		END IF;
		FOR R IN (SELECT * FROM LEADERBOARD WHERE CONTEST_ID=CID
ORDER BY STANDING)
		LOOP
			IF(FLAG=4)THEN
				EXIT;
			END IF;
			IF(LAST_STANDING <> R.STANDING)THEN
				LAST_STANDING := R.STANDING;
				FLAG := FLAG+ 1;
			END IF;

			IF(FLAG=1)THEN
				INSERT INTO HAS_MEDALS
VALUES(R.HANDLE,R.CONTEST_ID,'GOLD');
			ELSIF(FLAG=2)THEN
				INSERT INTO HAS_MEDALS
VALUES(R.HANDLE,R.CONTEST_ID,'SILVER');
			ELSIF(FLAG=3)THEN
				INSERT INTO HAS_MEDALS
VALUES(R.HANDLE,R.CONTEST_ID,'BRONZE');
			END IF;
		END LOOP;

		FOR R IN (SELECT * FROM HAS_MEDALS WHERE CONTEST_ID=CID)
		LOOP
			H_M.HANDLE := R.HANDLE;
			H_M.CONTEST_ID := R.CONTEST_ID;
			H_M.MEDAL_TYPE := R.MEDAL_TYPE;
			RETURN NEXT H_M;
		END LOOP;

		RETURN;
	END
$BODY$ LANGUAGE PLPGSQL;
```

--FUNCTION_9: UPDATE_RATING_AFTER_CONTEST

```
CREATE OR REPLACE FUNCTION UPDATE_RATING_AFTER_CONTEST(CID
INTEGER) RETURNS VOID
AS $BODY$
	DECLARE
		TOTAL_RATING INTEGER := 0;
		TOTAL_OPPONENTS INTEGER := 0;
		TOTAL_SCORE INTEGER := 0;
		OPPONENTS_RATING INTEGER := 0;
		HANDLE_RATING INTEGER := 0;
		WIN INTEGER := 0;
		LOSS INTEGER := 0;
		UPDATE_RATING INTEGER := 0;
		R RECORD;
	BEGIN
		SELECT SUM(MAXIMUM_SCORE) INTO TOTAL_SCORE FROM
QUESTIONS WHERE CONTEST_ID=123;
		RAISE NOTICE 'TOTAL SCORE = %',TOTAL_SCORE;

		SELECT COUNT(*) INTO TOTAL_OPPONENTS FROM leaderboard
WHERE CONTEST_ID=CID;
		TOTAL_OPPONENTS := TOTAL_OPPONENTS - 1;
		RAISE NOTICE 'TOTAL_OPPONENTS = %',TOTAL_OPPONENTS;

		SELECT SUM(RATING) INTO TOTAL_RATING FROM USERS WHERE
HANDLE IN (SELECT HANDLE FROM LEADERBOARD WHERE CONTEST_ID=CID);
		RAISE NOTICE 'TOTAL RATING = %',TOTAL_RATING;

		FOR R IN (SELECT * FROM LEADERBOARD WHERE CONTEST_ID=CID
ORDER BY STANDING)
		LOOP
			SELECT RATING INTO HANDLE_RATING FROM USERS WHERE
USERS.HANDLE=R.HANDLE;
			OPPONENTS_RATING := TOTAL_RATING - HANDLE_RATING;

			SELECT COUNT(*) INTO WIN FROM LEADERBOARD WHERE
CONTEST_ID=CID AND R.STANDING < LEADERBOARD.STANDING;
			SELECT COUNT(*) INTO LOSS FROM LEADERBOARD WHERE
CONTEST_ID=CID AND R.STANDING > LEADERBOARD.STANDING;

			UPDATE_RATING := (OPPONENTS_RATING +
(TOTAL_SCORE*(WIN-LOSS)))/TOTAL_OPPONENTS;
			IF(UPDATE_RATING<0) THEN
				UPDATE_RATING := 0;

			END IF;

			RAISE NOTICE 'WIN = % , LOSS = %',WIN,LOSS;
```

```
                        RAISE NOTICE 'INIT RATING = % , UPDATE RATING =
%',HANDLE_RATING,UPDATE_RATING;

                        UPDATE REGISTERS_IN SET
CONTEST_RATING=UPDATE_RATING WHERE CONTEST_ID=CID AND
HANDLE=R.HANDLE;

                END LOOP;
        END
$BODY$ LANGUAGE PLPGSQL;
```

--TRIGGER_1: INITIALISE USER RATING WITH 1200 ON REGISTER ON APPLICATION

```
CREATE OR REPLACE FUNCTION INIT_USER_RATING() RETURNS TRIGGER
AS $TRIGGER1$
        BEGIN
                IF (TG_OP = 'INSERT') THEN
                        NEW.RATING := 1200;
                        RAISE NOTICE 'INIT USER RATING WITH 1200';
                        RETURN NEW;
                END IF;
                RETURN NULL;
        END
$TRIGGER1$ LANGUAGE PLPGSQL;

CREATE TRIGGER TRIGGER1 BEFORE INSERT ON USERS
FOR EACH ROW EXECUTE PROCEDURE INIT_USER_RATING();
```

--TRIGGER_2: INITIALISE CONTEST_RATING WITH USER RATING ON INSERT ON REGISTRATION

```
CREATE OR REPLACE FUNCTION INIT_CONTEST_RATING() RETURNS TRIGGER
AS $TRIGGER2$
        DECLARE
                RATE INTEGER := 0;
        BEGIN
                IF (TG_OP = 'INSERT') THEN
                        SELECT RATING INTO RATE FROM USERS WHERE
USERS.HANDLE=NEW.HANDLE;
                        NEW.CONTEST_RATING := RATE;
                        RAISE NOTICE 'INIT CONTEST RATING WITH % USER RATING' ,
RATE;
```

```
                RETURN NEW;
            END IF;
            RETURN NULL;
        END
$TRIGGER2$ LANGUAGE PLPGSQL;

CREATE TRIGGER TRIGGER2 BEFORE INSERT ON REGISTERS_IN
FOR EACH ROW EXECUTE PROCEDURE INIT_CONTEST_RATING();
```

--TRIGGER_3: CHANGE USER RATING AFTER UPDATE CONTEST_RATING

```
CREATE OR REPLACE FUNCTION CHANGE_USER_RATING() RETURNS TRIGGER
AS $TRIGGER3$
        BEGIN
            IF (TG_OP = 'UPDATE') THEN
                UPDATE USERS SET RATING=NEW.CONTEST_RATING WHERE
USERS.HANDLE=NEW.HANDLE;
                RAISE NOTICE 'UPDATE USER RATING WITH % CONTEST
RATING' , NEW.CONTEST_RATING;
                RETURN NEW;
            END IF;
            RETURN NULL;
        END
$TRIGGER3$ LANGUAGE PLPGSQL;

CREATE TRIGGER TRIGGER3 AFTER UPDATE ON REGISTERS_IN
FOR EACH ROW EXECUTE PROCEDURE CHANGE_USER_RATING();
```