

## מיני פרויקט בבסיסי נתונים שלב ג

### Get Volunteer Count by Role

\*תיאור מילולי של התוכנית\* -

מקבלת מזהה תפקיד ומחזירה את מספר המתנדבים המשויכים get\_volunteer\_count\_by\_role הפונקציה סופרת את כמות Volunteers, לתפקיד זה. הפונקציה משתמשת בקורסור לקריאת רשומות מטבלת המתנדבים בתפקיד המבוקש ומחזירה את המספר הכולל. בנוסף, הפונקציה כוללת טיפול בחריגות.

```
CREATE OR REPLACE FUNCTION get_volunteer_count_by_role(p_role_id IN NUMBER)
RETURN NUMBER IS
    v_count NUMBER;
    CURSOR volunteer_cursor IS
        SELECT volunteer_ID
        FROM Volunteers
        WHERE role_ID = p_role_id;
    v_volunteer_id Volunteers.volunteer_ID%TYPE;
BEGIN
    v_count := 0;
    OPEN volunteer_cursor;
    LOOP
        FETCH volunteer_cursor INTO v_volunteer_id;
        EXIT WHEN volunteer_cursor%NOTFOUND;
        v_count := v_count + 1;
    END LOOP;
    CLOSE volunteer_cursor;

    RETURN v_count;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20001, 'An error occurred: ' || SQLERRM);
END;
```

## Get Total Shifts for Volunteer

- \*תיאור מילולי של התוכנית\*:

הפונקציה get\_total\_shifts\_for\_volunteer מקבלת מזהה מתנדב ומחזירה את מספר המשמרות הכולל שהמתנדב נרשם אליהן. הפונקציה משתמשת בקורסור לקריאת רשומות מטבלת signed\_up, סופרת את כמות המשמרות ומחזירה את המספר הכולל. בנוסף, הפונקציה כוללת טיפול בחריגות.

```
CREATE OR REPLACE FUNCTION get_total_shifts_for_volunteer(p_volunteer_id IN
NUMBER) RETURN NUMBER IS
    v_shift_count NUMBER := 0;
    CURSOR shift_cursor IS
        SELECT Shift_ID
        FROM signed_up
        WHERE Volunteer_ID = p_volunteer_id;
    v_shift_id signed_up.Shift_ID%TYPE;
BEGIN
    OPEN shift_cursor;
    LOOP
        FETCH shift_cursor INTO v_shift_id;
        EXIT WHEN shift_cursor%NOTFOUND;
        v_shift_count := v_shift_count + 1;
    END LOOP;
    CLOSE shift_cursor;

    RETURN v_shift_count;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'An error occurred: ' || SQLERRM);
END;
```

## Assign Volunteer to Shift

\*תיאור מילולי של התוכנית\*:

מקבלת מזהה מתנדב ומזהה משמרת, ומשייכת את המתנדב assign\_volunteer\_to\_shift הפרוצדורה למשמרת אם הוא לא משויך אליה כבר. הפרוצדורה משתמשת בקורסור לבדיקת השיוך הקיים, ומבצעת לשיוך חדש במידת הצורך. בנוסף, הפרוצדורה כוללת טיפול בחריגות INSERT פקודת

```
CREATE OR REPLACE PROCEDURE assign_volunteer_to_shift(p_volunteer_id IN
NUMBER, p_shift_id IN NUMBER) IS
    CURSOR check_cursor IS
        SELECT COUNT(*)
        FROM signed_up
        WHERE Volunteer_ID = p_volunteer_id AND Shift_ID = p_shift_id;
    v_count NUMBER;
BEGIN
    OPEN check_cursor;
    FETCH check_cursor INTO v_count;
    CLOSE check_cursor;

    IF v_count = 0 THEN
        INSERT INTO signed_up (Volunteer_ID, Shift_ID)
        VALUES (p_volunteer_id, p_shift_id);
        COMMIT;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Volunteer is already signed up for this shift.');
```

## Update Volunteer Role

- \*תיאור מילולי של התוכנית\*:

הפונקציה `update_volunteer_role` מקבלת מזהה מתנדב ומזהה תפקיד חדש, ומעדכנת את התפקיד של המתנדב לתפקיד החדש. הפונקציה כוללת בדיקה מקדימה לוודא שהמתנדב קיים, ולאחר מכן מבצעת פקודת `UPDATE` לעדכון התפקיד. בנוסף, הפונקציה כוללת טיפול בחריגות.

```
CREATE OR REPLACE PROCEDURE update_volunteer_role(p_volunteer_id IN NUMBER,
p_new_role_id IN NUMBER) IS
    v_old_role_id NUMBER;
BEGIN
    -- Check if volunteer exists
    SELECT role_ID
    INTO v_old_role_id
    FROM Volunteers
    WHERE volunteer_ID = p_volunteer_id;

    -- Update role
    UPDATE Volunteers
    SET role_ID = p_new_role_id
    WHERE volunteer_ID = p_volunteer_id;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Volunteer role updated from ' || v_old_role_id || '
to ' || p_new_role_id);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Volunteer not found. ');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20004, 'An error occurred: ' || SQLERRM);
END;
```

## Assign Volunteer and Get Count

\*תיאור מילולי של התוכנית\* -

לשיוך מתנדב למשמרת, ולאחר מכן assign\_volunteer\_to\_shift התוכנית הראשית קוראת לפרוצדורה כדי לקבל את מספר המשמרות הכולל של המתנדב get\_total\_shifts\_for\_volunteer קוראת לפונקציה ומדפיסה את התוצאה.

```
BEGIN
-- קריאה לפרוצדורה לשיוך מתנדב למשמרת
assign_volunteer_to_shift(1, 1);

-- קריאה לפונקציה לקבלת מספר המשמרות עבור המתנדב
DECLARE
v_shift_count NUMBER;
BEGIN
v_shift_count := get_total_shifts_for_volunteer(1);
DBMS_OUTPUT.PUT_LINE('Total shifts for volunteer 1: ' ||
v_shift_count);
END;
END;
```

## Update Role and Get Volunteer Count

\*תיאור מילולי של התוכנית\* -

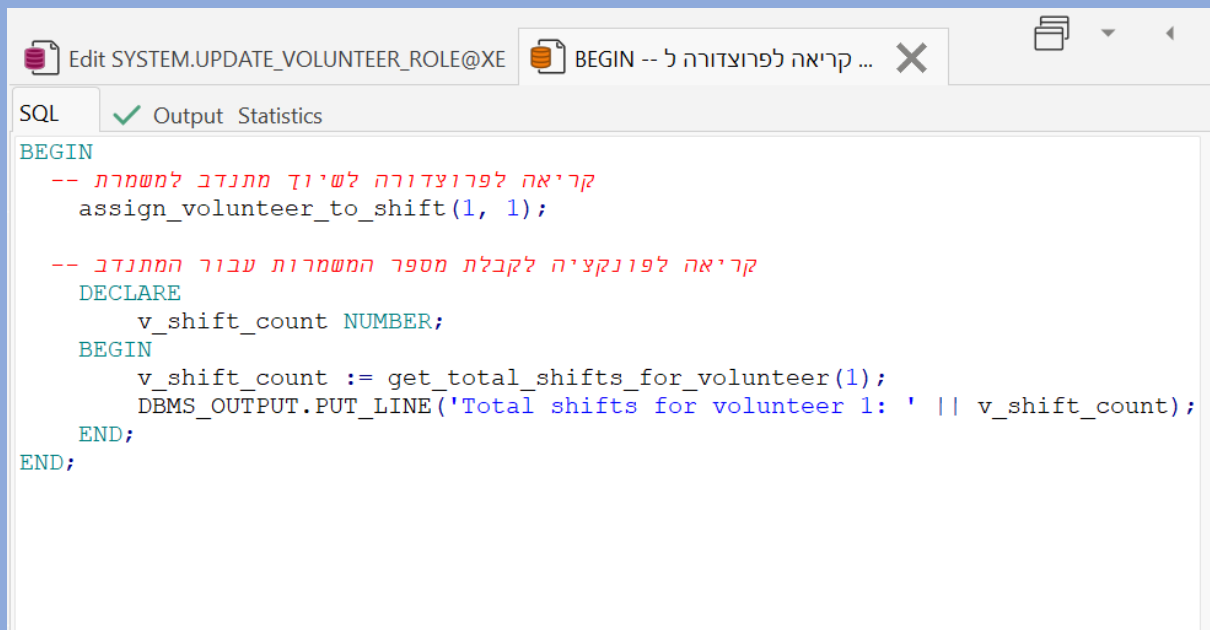
לעדכון התפקיד של מתנדב, ולאחר מכן update\_volunteer\_role התוכנית הראשית קוראת לפרוצדורה כדי לקבל את מספר המתנדבים המשויכים לתפקיד החדש get\_volunteer\_count\_by\_role קוראת לפונקציה ומדפיסה את התוצאה.

```
BEGIN
-- קריאה לפרוצדורה לעדכון תפקיד המתנדב
update_volunteer_role(1, 3);

-- קריאה לפונקציה לקבלת מספר המתנדבים בתפקיד
DECLARE
v_volunteer_count NUMBER;
BEGIN
v_volunteer_count := get_volunteer_count_by_role(2);
DBMS_OUTPUT.PUT_LINE('Total volunteers in role 2: ' ||
v_volunteer_count);
END;
```

END;

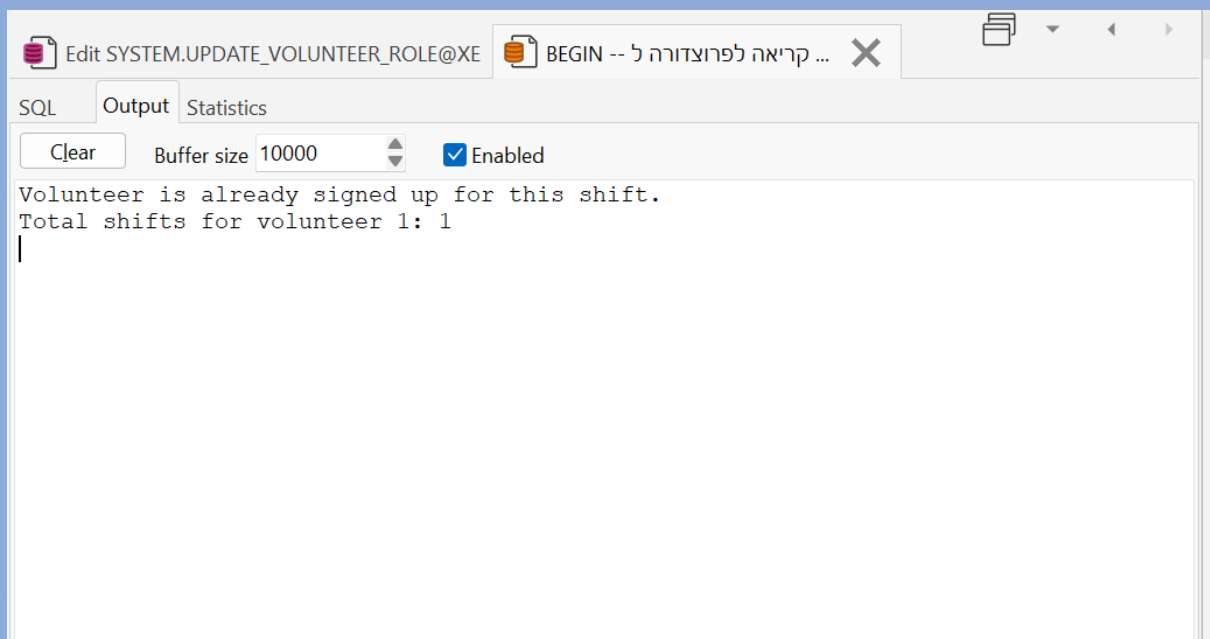
הוכחת הרצה MAIN ראשון:



The screenshot shows a SQL editor window with two tabs: "Edit SYSTEM.UPDATE\_VOLUNTEER\_ROLE@XE" and "BEGIN -- קריאה לפרוצדורה ל...". The "SQL" tab is active, displaying the following SQL script:

```
BEGIN
-- קריאה לפרוצדורה לשיוך מתנדב למשמרת
assign_volunteer_to_shift(1, 1);

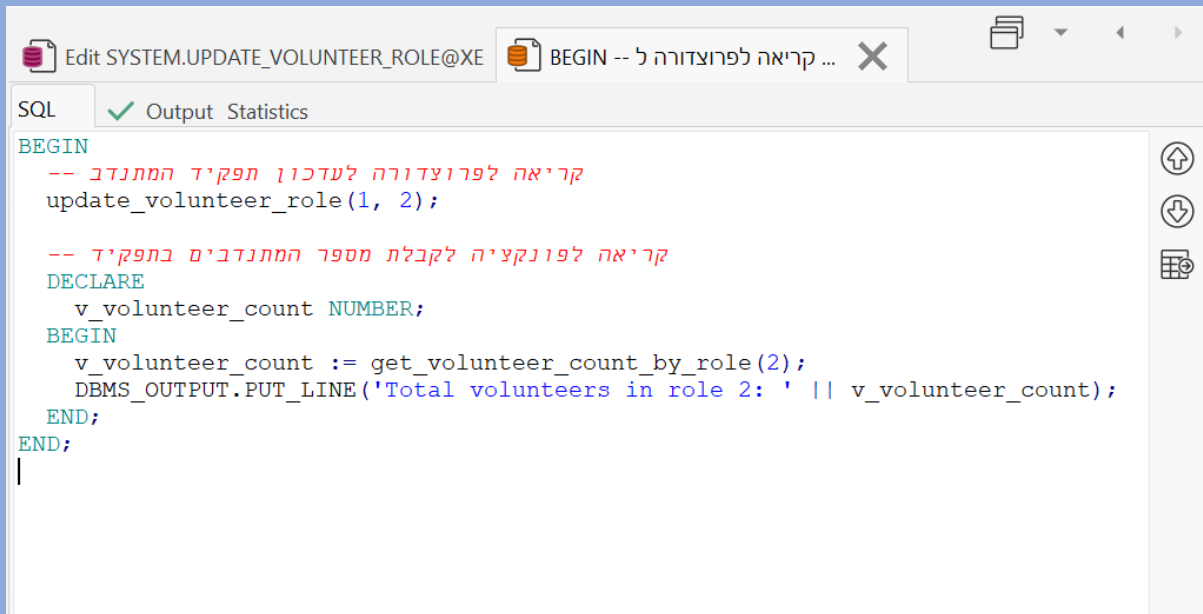
-- קריאה לפונקציה לקבלת מספר המשמרות עבור המתנדב
DECLARE
    v_shift_count NUMBER;
BEGIN
    v_shift_count := get_total_shifts_for_volunteer(1);
    DBMS_OUTPUT.PUT_LINE('Total shifts for volunteer 1: ' || v_shift_count);
END;
END;
```



The screenshot shows the same SQL editor window, but the "Output" tab is active. It displays the results of the SQL script execution:

```
Volunteer is already signed up for this shift.
Total shifts for volunteer 1: 1
|
```

## הוכחת הרצה MAIN שני:

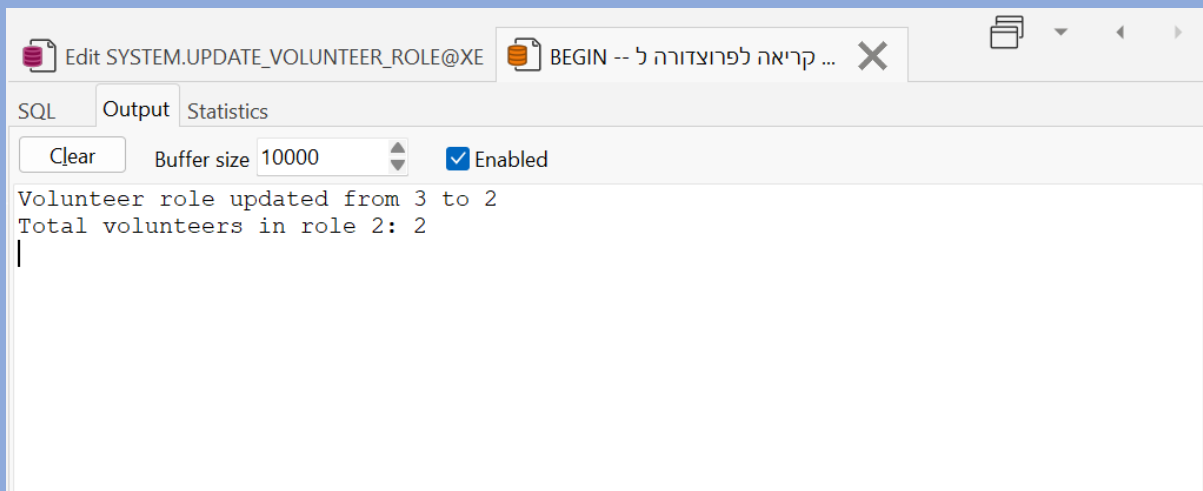


The screenshot shows the SQL Developer interface with the following components:

- Top Bar:** Contains the file name "Edit SYSTEM.UPDATE\_VOLUNTEER\_ROLE@XE" and a tab titled "BEGIN -- קריאה לפרוצדורה ל ...".
- SQL Editor:** Displays the following PL/SQL code:

```
BEGIN
  -- קריאה לפרוצדורה לעדכון תפקיד המתנדב
  update_volunteer_role(1, 2);

  -- קריאה לפונקציה לקבלת מספר המתנדבים בתפקיד
  DECLARE
    v_volunteer_count NUMBER;
  BEGIN
    v_volunteer_count := get_volunteer_count_by_role(2);
    DBMS_OUTPUT.PUT_LINE('Total volunteers in role 2: ' || v_volunteer_count);
  END;
END;
```
- Right Panel:** Includes icons for running the script (a green play button), saving, and other utility functions.



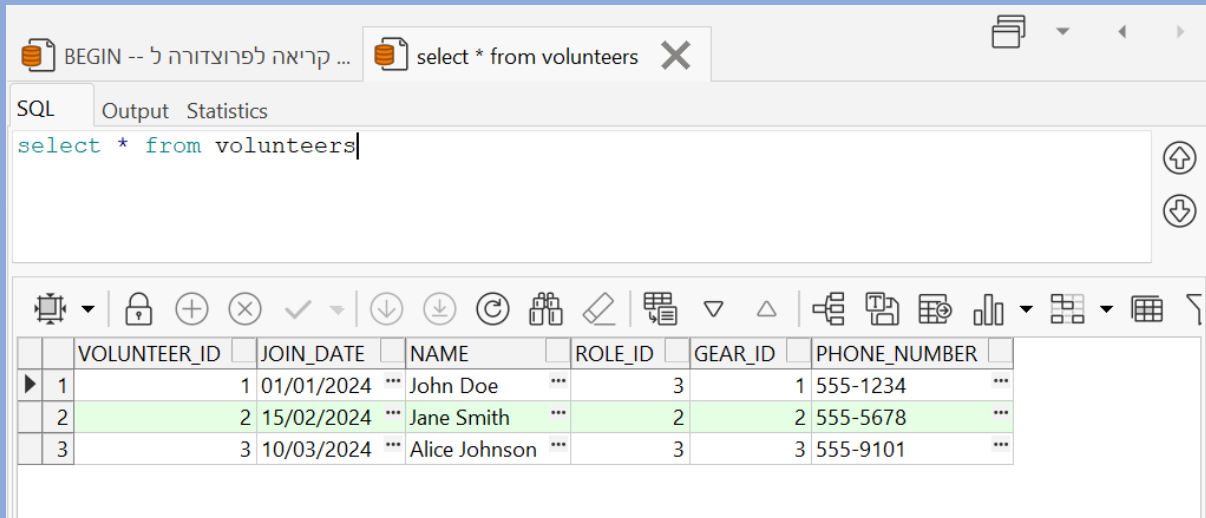
The screenshot shows the SQL Developer interface with the following components:

- Top Bar:** Same as the previous screenshot, showing the file name and the "BEGIN -- קריאה לפרוצדורה ל ..." tab.
- Output Panel:** The "Output" tab is selected, showing the results of the script execution:

```
Volunteer role updated from 3 to 2
Total volunteers in role 2: 2
```
- Statistics Panel:** The "Statistics" tab is also visible, showing execution statistics.

הוכחה לשינוי בבסיס הנתונים:

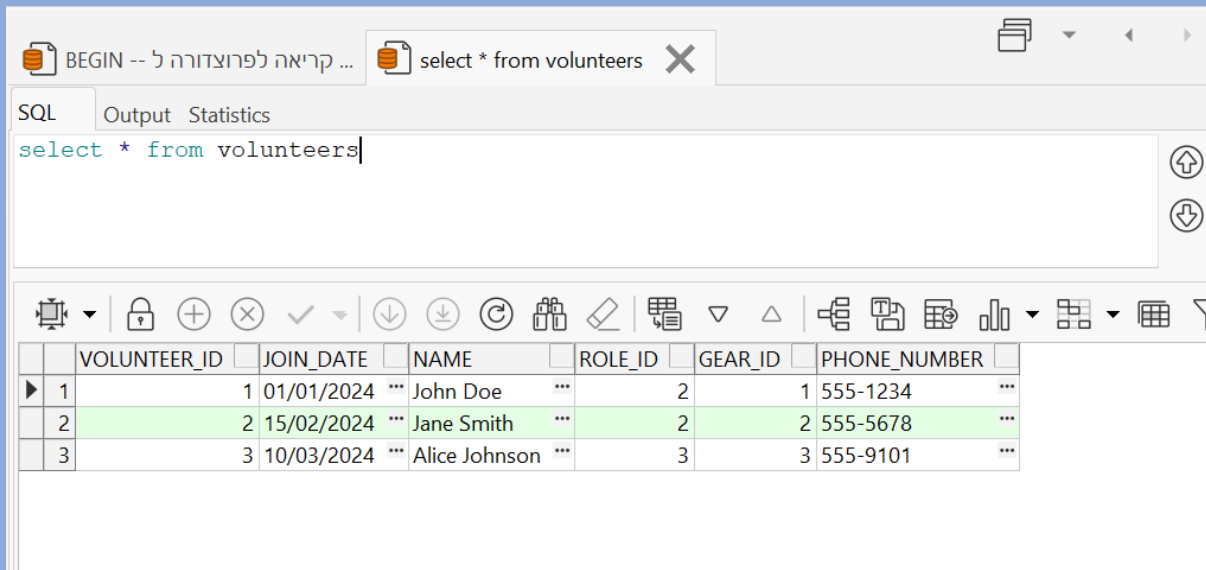
לפני:



The screenshot shows a SQL IDE window with a tab titled "BEGIN -- ל פרוצדורה ל ...". The SQL editor contains the query `select * from volunteers`. The output pane displays a table with 8 columns: VOLUNTEER\_ID, JOIN\_DATE, NAME, ROLE\_ID, GEAR\_ID, PHONE\_NUMBER, and two unnamed columns. The data is as follows:

	VOLUNTEER_ID	JOIN_DATE	NAME	ROLE_ID	GEAR_ID	PHONE_NUMBER		
▶ 1	1	01/01/2024	John Doe	...	3	1 555-1234	...	
2	2	15/02/2024	Jane Smith	...	2	2 555-5678	...	
3	3	10/03/2024	Alice Johnson	...	3	3 555-9101	...	

אחרי:



The screenshot shows the same SQL IDE window after a change. The query `select * from volunteers` still runs, but the output table now shows a different set of data:

	VOLUNTEER_ID	JOIN_DATE	NAME	ROLE_ID	GEAR_ID	PHONE_NUMBER		
▶ 1	1	01/01/2024	John Doe	...	2	1 555-1234	...	
2	2	15/02/2024	Jane Smith	...	2	2 555-5678	...	
3	3	10/03/2024	Alice Johnson	...	3	3 555-9101	...	

סיכום הדו"ח:

כל התוכניות שנכתבו כוללות טיפול בחריגות, שימוש בקורסורים (גם מפורשים וגם רמוזים), פקודות DML, לולאות, והסתעפויות לפי הנדרש. הדו"ח כולל תיאור מפורט של כל פונקציה ופרוצדורה, הקוד שלהן, והוכחה שהן אכן עובדות ומבצעות את הנדרש מהן ללא תקלות.