### דו"ח שלב 3

213385263 - **מגישות:** תהילה בן משה מרים פרץ - 322241100

### <u>פונקציות:</u>

פונקציה מס' 1

<u>תיאור:</u> בהינתן מזהה מדריך, נרצה לחשב את הבונוס שיקבל לאחר סיום הקורס: כמות ההרשמה לקורס שמלמד \* מחיר הקורס \*10%

### <u>הקוד:</u>

```
1 🖯 create or replace function ReturnInstractorSalary(instructorID in number) return number is
 2 🖨
       Result NUMBER := 0;
        registrationCount INTEGER;
 3
        CURSOR courseCursor IS
 4 🖨
 5 🖨
            SELECT C.Course_ID, C.price
            FROM Course C
 6
            WHERE C.Instructor ID = instructorID;
 7
 8
        courseRecord courseCursor%ROWTYPE;
 9
    BEGIN
10
11
        -- Open the cursor
        OPEN courseCursor;
12
13
14
        -- Loop through each course taught by the instructor
15 卓
16
            -- Fetch the next course
            FETCH courseCursor INTO courseRecord;
17
            EXIT WHEN courseCursor%NOTFOUND;
18
19
20
            -- Get the number of registrations for the current course
21 🖨
22 🖨
                SELECT COUNT (*)
23
                INTO registrationCount
24
                FROM Registration R
25
                WHERE R.Course_ID = courseRecord.Course_ID;
26
            EXCEPTION
               WHEN NO DATA FOUND THEN
27 🖨
28
                    registrationCount := 0;
            END;
29
30
31
             -- Calculate the salary contribution from the current course
32
             Result := Result + (registrationCount * courseRecord.price * 0.1);
        END LOOP;
33
34
35
         -- Close the cursor
        CLOSE courseCursor;
36
37
         -- Return the calculated salary
 39
         RETURN Result:
    END ReturnInstractorSalary;
 40
41
```

# לצורך הדוגמא ניקח את המדריך עם ת"ז = 113110911 ובקורס אצלו יש 6 משתתפים

```
--help query for function 1

SELECT DISTINCT I.Instructor_ID, I.fname, I.lname, C.Course_Name, C.PRICE
FROM Instructor I

JOIN Course C ON I.Instructor_ID = C.Instructor_ID

JOIN Registration R ON C.Course_ID = R.Course_ID

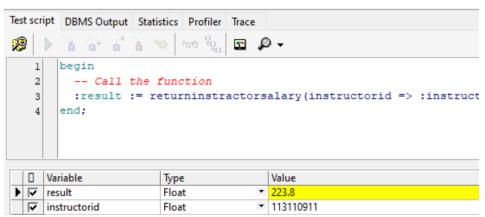
GROUP BY I.Instructor_ID, I.fname, I.lname, C.Course_Name, C.PRICE

HAVING COUNT (DISTINCT R.Participant_ID) = 6;

INSTRUCTOR_ID FNAME LNAME COURSE_NAME PRICE

I 113110911 Dave Mraz First Aid for Animal Bite 373
```

# לאחר הרצת הפונקציה:



## פונקציה מס' 2

#### תיאור:

בהינתן שם של מחסן קטגוריה (category location) נרצה לחשב על הערך המשוער של המחסן. הערך המשוער מחושב כך:

סכום האביזרים במחסן \* מספר הקטגוריות הנמצאות בו / המשקל הממוצע של האביזרים המאוחסנים במחסן.

הקוד:

```
CREATE OR REPLACE FUNCTION Calculate_Location_Value (p_category_location VARCHAR2)
RETURN NUMBER
IS
  -- Declare variables and types
 TYPE tool_rec IS RECORD (
   Tool_name Tool.Tool_name%TYPE,
   weight Tool.weight%TYPE
  TYPE tool_cursor IS REF CURSOR;
  v tool tool rec;
  v_tool_cursor tool_cursor;
  v_total_weight FLOAT := 0;
  v tool count NUMBER := 0;
  v_category_count NUMBER := 0;
  v_avg_weight FLOAT := 0;
  v_location_value FLOAT := 0;
  -- Declare implicit cursor
  CURSOR category_cursor IS
    SELECT Category_name
    FROM TCategory
    WHERE Category location = p category location;
  -- Get the number of categories in the location
  OPEN category_cursor;
  LOOP
   FETCH category_cursor INTO v_tool.Tool_name;
    EXIT WHEN category_cursor%NOTFOUND;
   v category count := v category count + 1;
   -- Explicit cursor to get tools for each category
   OPEN v_tool_cursor FOR
     SELECT Tool_name, weight
     FROM Tool
    WHERE Category_name = v_tool.Tool_name;
     FETCH v_tool_cursor INTO v_tool;
     EXIT WHEN v tool cursor%NOTFOUND;
     v total weight := v total weight + v tool.weight;
    v_tool_count := v_tool_count + 1;
   END LOOP;
   CLOSE v_tool_cursor;
 END LOOP;
 CLOSE category_cursor;
 -- Calculate the average weight
 IF v tool count > 0 THEN
   v_avg_weight := v_total_weight / v_tool_count;
 ELSE
   v_avg_weight := 0;
 END IF;
 -- Calculate the location value
 IF v_avg_weight > 0 THEN
   v location value := (v category count * v tool count) / v avg weight;
 ELSE
   v_location_value := 0;
 END IF;
```

```
-- Return the calculated location value

RETURN v_location_value;

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('No data found for the given category location.');

RETURN 0;

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);

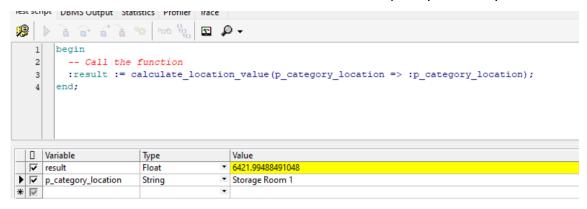
RETURN 0;

END Calculate_Location_Value;
```

### <u>:דוגמא</u>

לצורך הדוגמא ניקח את המחסן 'Storage room 1'

נחשב את הערך של המחסן בפונקציה:



לצורך בדיקה, נכתוב 3 שאילתות. אחת תחזיר את כמות האביזרים במחסן, אחת את מספר הקטגוריות בו ואחת את המשקל הממוצע:

נחשב לפי הנוסחה שלעיל ונקבל מספר השווה לתוצאת הפונקציה שלנו.

### פרוצדורות:

#### פרוצדורה 1

### <u>תיאור:</u>

נרצה לבנות פרוצדורה שתעביר את כל המשתתפים מקורס אחד לשני עקב סגירת הקורס או העברתו למקום אחר. הפרוצדורה תעדכן את ההרשמה של המשתתפים בהתאם וכן תוודא שאין רישום כפול של המשתתף לקורס אליו הוא מועבר.

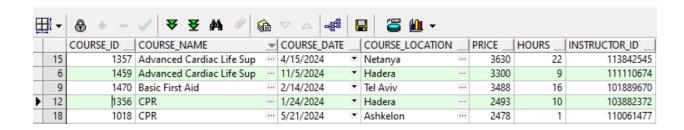
### <u>הקוד</u>:

```
CREATE OR REPLACE PROCEDURE TransferParticipants(
    from course id IN Course.Course ID%TYPE,
    to course id IN Course.Course ID%TYPE
IS
    TYPE ParticipantRec IS RECORD (
        participant id Participant.Participant ID%TYPE,
        registration id Registration.Registration ID%TYPE
    CURSOR participant_cursor IS
        SELECT p.Participant ID, r.Registration ID
        FROM Participant p
        JOIN Registration r ON p.Participant ID = r.Participant ID
        WHERE r.Course ID = from course id;
    participant_rec ParticipantRec;
    err_msg VARCHAR2(32767);
BEGIN
    OPEN participant cursor;
        FETCH participant cursor INTO participant rec;
        EXIT WHEN participant cursor%NOTFOUND;
```

```
-- Check if participant is already registered for the to course
       DECLARE
           duplicate_count INT;
       BEGIN
           SELECT COUNT (*)
            INTO duplicate_count
           FROM Registration
            WHERE Participant ID = participant rec.participant id
           AND Course_ID = to_course_id;
            IF duplicate_count = 0 THEN
                -- Update registration
                UPDATE Registration
                SET Course_ID = to_course_id
               WHERE Registration_ID = participant_rec.registration_id;
                -- Skip this participant
               DBMS OUTPUT.PUT LINE('Participant ' || participant rec.participant id || ' is already register
            END IF;
       END:
   END LOOP;
   CLOSE participant_cursor;
   OPEN ref cur FOR SELECT 'Participants transferred successfully' AS message FROM dual;
   COMMIT;
EXCEPTION
    WHEN OTHERS THEN
       err_msg := 'Error occurred: ' || TO_CHAR(SQLCODE) || ' - ' || SUBSTR(SQLERRM, 1, 200);
       OPEN ref cur FOR SELECT err msg AS message FROM dual;
       ROLLBACK:
END;
```

### <u>בדיקה:</u>

לצורך הבדיקה, מתוך רשימת הקורסים נמצא שני קורסים עם אותו שם- ניקח את CPR אך עם מיקום שונה ואותם נרצה לאחד:



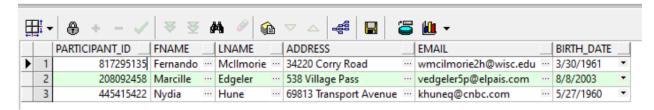
### כמות המשתתפים שיש בקורס הראשון '1356':

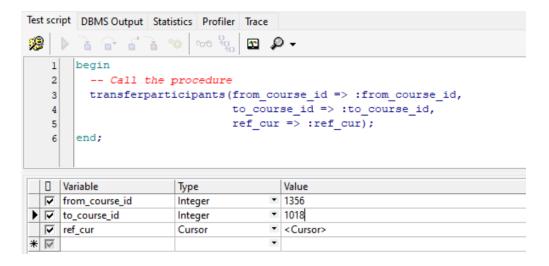
```
SELECT P.Participant_ID, P.fname, P.lname, P.address, P.email, P.birth_date
FROM Participant P
JOIN Registration R ON P.Participant_ID = R.Participant_ID
WHERE R.Course_ID = '1356';
```

			PARTICIPANT_ID	FNAME _	LNAME		ADDRESS		EMAIL		BIRTH_DATE	
Ī		1	631311297	Ozzy ··	Roj		983 Oakridge Avenue		troj4n@jigsy.com		12/27/2003	•
		2	883547761	Gannon ··	Matevushe	v	545 Transport Trail		rmatevushevb2@bandcamp.com		3/31/1966	-
		3	967655547	Ingaborg	Bedberry		3 Grayhawk Terrace		lbedberrya9@google.co.jp		5/14/1963	•

### בקורס השני '1018':

```
SELECT P.Participant_ID, P.fname, P.lname, P.address, P.email, P.birth_date
FROM Participant P
JOIN Registration R ON P.Participant_ID = R.Participant_ID
WHERE R.Course ID = '1018';
```





לאחר הפעלת הפרוצדורה, בקורס הראשון ניתן לראות שאין משתתפים:

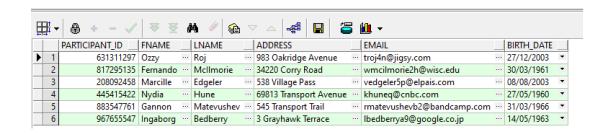
### ואכן המשתתפים עברו לקורס השני:

```
----- 1356 1018

SELECT P.Participant_ID, P.fname, P.lname, P.address, P.email, P.birth_date
FROM Participant P

JOIN Registration R ON P.Participant_ID = R.Participant_ID

WHERE R.Course_ID = '1018';
```



#### פרוצדורה 2

#### <u>תיאור:</u>

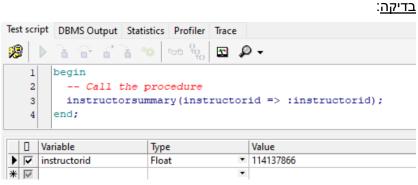
בהיתן מזהה מדריך שידוע על הצטיינותו, נרצה לבנות פרוצדורה שתחזיר סיכום מלא עם פרטיו. הפרטים כוללים: שמות ותאריכי הקורסים שמלמד וכן המשתתפים בכל קורס. הפרוצדורה תוסיף טיפול בחריגות במקרה שבו מוכנס מזהה שגוי .

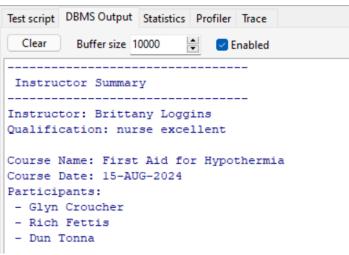
בנוסף, הפרוצדורה תעדכן בבסיס הנתונים את הסמכת המדריך - תוסיף את המילה "excellent" לצד הסמכתו הרגילה.

### <u>הקוד</u>:

```
1 CREATE OR REPLACE PROCEDURE InstructorSummary (InstructorID IN NUMBER) IS
     -- Cursor to fetch the courses taught by the instructor
3 🖨
     CURSOR courses_cur IS
       SELECT c.Course ID, c.Course name, c.Course date
5
       FROM Course c
       WHERE c.Instructor ID = InstructorID
 6
7
       ORDER BY c.Course_date DESC;
8
      -- Variable to hold each fetched row from courses cur
9
.0
     course_record courses_cur%ROWTYPE;
1
.2
      -- Variables to hold instructor details
     v_instructor_fname Instructor.fname%TYPE;
.3
.4
     v_instructor_lname Instructor.lname%TYPE;
     v_instructor_qualification Instructor.qualification%TYPE;
.6
.7
     -- Variable to check if instructor exists
.8
     v instructor exists NUMBER := 0;
9
    BEGIN
     -- Increase the buffer size to 1,000,000 bytes
:0
      DBMS_OUTPUT.ENABLE(1000000);
:1
2
:3
       -- Fetch instructor's name and qualification
     SELECT fname, lname, qualification INTO v_instructor_fname, v_instructor_lname, v_instructor_qualification
:4 🖨
5
      FROM Instructor
6
      WHERE Instructor_ID = InstructorID;
28
      -- Check if instructor exists
      SELECT COUNT (*)
29 🛱
      INTO v_instructor_exists
30
31
      FROM Instructor
32
      WHERE Instructor ID = InstructorID;
33
34 🖨
      IF v instructor exists = 0 THEN
       DBMS_OUTPUT.PUT_LINE('Instructor with ID ' || InstructorID || ' does not exist.');
35
36
        RETURN:
      END IF;
37
38
39
      -- Enhance qualification with "excellent"
40
      v_instructor_qualification := v_instructor_qualification || ' excellent';
41
42
      -- Update instructor's qualification in the database
43 白
      UPDATE Instructor
44
      {\tt SET \ qualification = v\_instructor\_qualification}
      WHERE Instructor_ID = InstructorID;
45
46
47
      -- Commit the update
48
      COMMIT:
49
50
      DBMS_OUTPUT.PUT_LINE('----');
      DBMS OUTPUT.PUT LINE(' Instructor Summary');
51
      DBMS OUTPUT.PUT LINE('----');
52
      DBMS_OUTPUT.PUT_LINE('Instructor: ' || v_instructor_fname || ' ' || v_instructor_lname);
53
      DBMS OUTPUT.PUT LINE('Qualification: ' || v instructor qualification);
54
55
      DBMS OUTPUT.PUT LINE('');
```

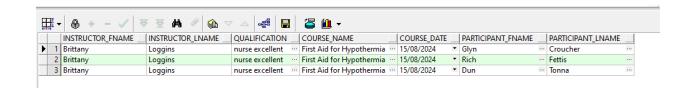
```
-- Open the cursor for courses
 OPEN courses_cur;
 T.OOP
   FETCH courses cur INTO course record;
   EXIT WHEN courses cur%NOTFOUND;
   -- Print course details
   DBMS OUTPUT.PUT LINE('Course Name: ' || course record.Course name);
   DBMS_OUTPUT.PUT_LINE('Course Date: ' || TO_CHAR(course_record.Course_date, 'DD-MON-YYYY'));
   DBMS OUTPUT.PUT LINE('Participants:');
   -- Cursor to fetch the participants for the current course
   FOR participant record IN (
     SELECT p.fname, p.lname
     FROM Participant p
     JOIN Registration r ON r.Participant_ID = p.Participant_ID
    WHERE r.Course_ID = course_record.Course_ID
    ORDER BY p.lname, p.fname
   ) LOOP
     -- Print participant details
     DBMS_OUTPUT.PUT_LINE(' - ' || participant_record.fname || ' ' || participant_record.lname);
   END LOOF;
   DBMS OUTPUT.PUT LINE('');
 END LOOP;
 CLOSE courses_cur;
EXCEPTION
 WHEN NO DATA FOUND THEN
  DBMS_OUTPUT.PUT_LINE('No data found for this Instructor ' || InstructorID);
 WHEN OTHERS THEN
  DBMS OUTPUT.PUT LINE('An error occurred: ' || SQLERRM);
END InstructorSummary;
```





נוודא שהפרטים עבור המדריך שהכנסנו נכונים. לצורך כך נכתוב שאילתא שתחזיר את פרטי המדריך עבור המזהה שהכנסנו לעיל (כולל העדכון שנעשה בבסיס הנתונים):

```
SELECT
   i.fname AS Instructor fname,
   i.lname AS Instructor lname,
   i.qualification ,
   c.Course_name,
   c.Course_date,
   p.fname AS Participant fname,
   p.lname AS Participant lname
FROM
   Course c
JOIN
   Registration r ON c.Course_ID = r.Course_ID
JOIN
   Participant p ON r.Participant ID = p.Participant ID
JOIN
   Instructor i ON c.Instructor_ID = i.Instructor_ID
WHERE
   c.Instructor ID = 114137866
ORDER BY
   c.Course name,
   c.Course date,
   p.lname,
   p.fname;
```



### תוכניות ראשיות

#### תכנית 1:

בתכנית נקבל קלט מהמשתמש של מזהה מדריך. נרצה לקשר בין הפונקציה שמחשבת את הבונוס של המדריך לבין הפרוצדורה שמחזירה את הדו"ח הכולל את הקורסים שמלמד והמשתתפים בהם.

לצורך כך, נוסיף בטבלת המדריך רשומה של instructor\_salary אשר תכיל בתוכה ערכים של משכורת המדריך. התכנית הראשית תרצה לקבל רק את פרטי המדריכים המצטיינים. מדריך מצטיין = בונוס >200 לצורך כך היא תקרא לפונקציה של הבונוס עבור כל המדריכים בבסיס הנתונים ועבור כל מדריך מצטיין- היא תוסיף את הבונוס למשכורת הכוללת ותקרא לפרוצדורה שתדפיס את הדו"ח הכולל. יחד עם דו"ח זה תודפס גם המשכורת.

### עדכוו ההוספה:

```
SQL Output Statistics

ALTER TABLE Instructor
ADD instructor_salary NUMERIC(7, 2);

BEGIN

UPDATE Instructor
SET instructor_salary = ROUND(2000 + (6000 - 2000) * DBMS_RANDOM.VALUE, 2);
COMMIT;
END;
```

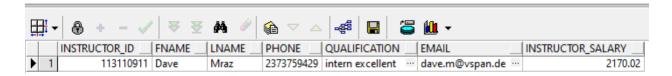
### <u>התכנית:</u>

```
Test script DBMS Output Statistics Profiler Trace
🥦 | ▶ 🚡 🔐 🚰 🐃 | ∞ % | 🖬 🗩 ▾
       DECLARE
   1
           CURSOR instructorCursor IS
               SELECT Instructor ID
               FROM Instructor;
           i instructorid Instructor.Instructor ID%TYPE;
                       NUMBER; -- Variable to hold the function result
             - Open the cursor to iterate over each instructor
  10
           OPEN instructorCursor;
  11
  12
  13
               -- Fetch the next instructor
               FETCH instructorCursor INTO i_instructorid;
  14
  15
               EXIT WHEN instructorCursor%NOTFOUND;
  16
  17
               -- Call the function and store the result
  18
              i_bonus := ReturnInstractorSalary(instructorid => i_instructorid);
  19
  20
               -- Check if the bonus is greater than 200
  21
               IF i_bonus > 200 THEN
                     -
-- Update the instructor's salary
  22
  23
                   UPDATE Instructor
  24
                   SET instructor_salary = instructor_salary + i_bonus
                   WHERE Instructor_ID = i_instructorid;
  25
  26
  27
                   -- Call the procedure
                   InstructorSummary(instructorid => i instructorid);
  28
  29
                    -- Display the salary result with the instructor report
  30
                   DBMS OUTPUT.PUT LINE ('Instructor ID: ' || i instructorid || ' - Bonus Salary: ' || i bonus);
  31
               END IF;
  32
           END LOOP;
  33
  34
  35
            -- Close the cursor
  36
           CLOSE instructorCursor;
   37
            -- Commit the changes
           COMMIT;
   38
       END;
```

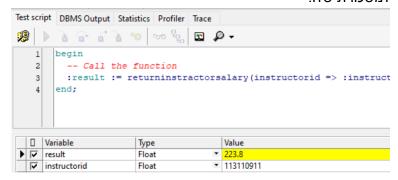
### נבדוק שהתכנית אכן עבדה:

עבור המדריך הספציפי, לפני הרצת התכנית הראשית, כך נראתה המשכורת שלו.

SELECT \*
FROM Instructor
WHERE Instructor ID = 113110911;

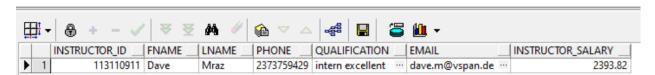


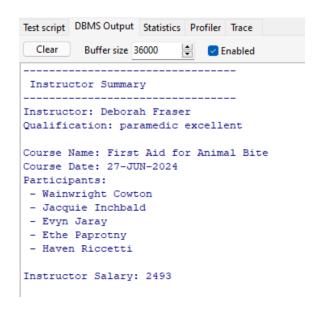
הרצת הפונקציה החזירה בונוס של 223.8 . מכיוון שערך זה >200 המדריך נחשב כמצטיין והבונוס יתווסף למשכורת שלו.



לאחר הרצת התכנית הראשית ניתן לראות שהבונוס נוסף:

SELECT \*
FROM Instructor
WHERE Instructor\_ID = 113110911;





### :2 תכנית

בתכנית נרצה לקרוא לפונקציה שמחשבת את ערך המחסן וכן לפרוצדורה שמעבירה משתתפים מקורס אחד לשני.

כדי שיהיה קשר בין השנים, התכנית תקבל שני ערכים של מזהה קורס כדי להעביר את המשתתפים, לאחר מכן היא תבדוק באיזה אביזרים הקורס השני משתמש ובאיזה מחסנים הם נמצאים. עבור כל מחסן כזה היא תחשב את ערכו ותדפיס.

### <u>הקוד:</u>

```
DECLARE
                    BOOLEAN := TRUE; -- Flag to check if the procedure worked
 2
       v success
      v_function_res NUMBER;
                                         -- Variable to hold the function result
 3
      v error message VARCHAR2(4000); -- Variable to hold any error message
 4
 5
 6
       -- Define a cursor to fetch tools in the destination course's warehouse
 7
      CURSOR tool cursor IS
         SELECT t. Tool name, tc. Category location
 8
        FROM Tool t
 9
        JOIN Course tool ct ON t.Tool name = ct.Tool name
 10
 11
        JOIN TCategory to ON t.Category_name = tc.Category_name
        WHERE ct.Course_ID = :to_course_id;
 12
 13
      v_tool tool_cursor%ROWTYPE; -- Variable to hold each fetched tool
 14
15
    BEGIN
 16
      DBMS OUTPUT.ENABLE (500000);
 17
 18
 19
       -- Call the procedure to transfer participants
     TransferParticipants(from_course_id => :from_course_id, to_course_id => :to_course_id);
 20
     DBMS OUTPUT.PUT LINE('Participants transferred successfully.');
 21
 22
23
     -- Open the cursor to fetch tools in the destination course's warehouse
     OPEN tool cursor;
24
25
26
       -- Fetch the next tool
27
28
       FETCH tool_cursor INTO v_tool;
29
      EXIT WHEN tool_cursor%NOTFOUND;
30
31
       -- Call the function to calculate the tool's location value
32
      v_function_res := Calculate_Location_Value(p_category_location => v_tool.Category_location);
33
34
       -- Output the function result for each tool
      DBMS_OUTPUT.PUT_LINE('Tool: ' || v_tool.Tool_name || ', Category Location: ' || v_tool.Category_location
35
     END LOOP;
36
37
38
      -- Close the cursor
     CLOSE tool cursor;
39
40
41
     -- Commit the changes
     COMMIT;
42
43
   EXCEPTION
44
     WHEN OTHERS THEN
45
       v_success := FALSE;
46
       v_error_message := SQLERRM;
47
       DBMS_OUTPUT.PUT_LINE('Error: ' || v_error_message);
48
49
       ROLLBACK;
   END;
50
```

# <u>הפלט:</u>

