

שלב 3 - תוכניות -

- א. 2 פונקציות.
ב. 2 פרוצדורות.
ג. 2 תוכניות ראשיות המזמנות כל אחת פונקציה אחת ופרוצדורה אחת.
-

תוכנית 1

- DECLARE
- order_id_1 NUMBER;
- TotalP NUMBER;
- loc NUMBER;
- orderCursor SYS_REFCURSOR;
- BEGIN
- --בודק את המיקום של ההזמנה
- loc := OrderChecker;
- --מחזיר את כל ההזמנות במיקום הזה
- orderCursor := get_order(loc);
-
- LOOP
- --עוברת על כל התז של ההזמנות
- FETCH orderCursor INTO order_id_1;
- EXIT WHEN orderCursor%NOTFOUND;
-
- -- מחשבת את המחיר הסופי
- TotalP := get_order_cost(order_id_1);
-
- --עושה הנחה ומעדכנת את המחיר הסופי
- Discount(order_id_1,PlatoT,1
- END LOOP;
- CLOSE orderCursor;
- END;

get_order_cost -

- CREATE OR REPLACE FUNCTION
get_order_cost(p_orderID NUMBER)
RETURN number IS
- v_total_cost NUMBER;
- BEGIN
- -- המחיר הסופי הוא כמות הכרטיסים כפול המחיר של
הכרטיס
- SELECT ticketAmount * ticketCost INTO
v_total_cost
- FROM Orders
- WHERE orderID = p_orderID;
-
- RETURN v_total_cost;
- END;

הסבר get_order_cost -

הגדרת פונקציה: הפונקציה מוגדרת עם השם `get_order_cost` ומקבלת פרמטר אחד `p_orderID` מסוג מספר (NUMBER).

משתנה מקומי: בתוך הפונקציה מוגדר משתנה מקומי `v_total_cost` מסוג מספר (NUMBER), שימש לאחסון העלות הכוללת.

שאילתת SQL: השאילתה משתמשת בפרמטר `p_orderID` כדי לבחור את כמות הכרטיסים (`ticketAmount`) ואת המחיר של הכרטיס (`ticketCost`) מהטבלה `Orders`, מחשבת את העלות הכוללת על ידי הכפלת כמות הכרטיסים במחיר הכרטיס, ושומרת את התוצאה במשתנה `v_total_cost`.

החזרת תוצאה: הפונקציה מחזירה את הערך של `v_total_cost` כעלות הכוללת של ההזמנה.

OrderChecker

```
1 create or replace function OrderChecker
2
3     RETURN NUMBER IS
4     v_orderID NUMBER;
5 BEGIN
6
7     SELECT locationId INTO v_orderID
8     FROM (SELECT locationId FROM Locations ORDER BY DBMS_RANDOM.RANDOM)
9     WHERE ROWNUM = 1;
10
11     RETURN v_orderID;
12 END;
```

הסבר OrderChecker-

הפונקציה מוגדרת להחזיר ערך מסוג מספר (`NUMBER`).

משתנה מקומי בשם `v_orderID` מוגדר לאחסון התוצאה.

בתוך הבלוק של ה-`BEGIN`:

מתבצע שאילתת SQL כדי לבחור מזהה מיקום אקראי מהטבלה `Locations`.

זה נעשה על ידי שימוש ב-`DBMS_RANDOM.RANDOM` כדי לערבב את רשומות הטבלה ולבחור את

הרשומה הראשונה (באמצעות `ROWNUM = 1`).

התוצאה נשמרת במשתנה `v_orderID`.

הפונקציה מחזירה את הערך של `v_orderID`.

בפועל, הפונקציה בוחרת מזהה מיקום אקראי מטבלת ה-`Locations` ומחזירה אותו.

get_order

parameter list

```
1 CREATE OR REPLACE FUNCTION get_order(p_c_id IN NUMBER)
2 RETURN SYS_REFCURSOR IS
3     orderList SYS_REFCURSOR;
4 BEGIN
5     OPEN orderList FOR
6         SELECT orderID
7         FROM   orderID join eventID join Locations
8         WHERE  b.c_id = p_c_id;
9
10
11
12
13     CLOSE bookingsCursor;
14
15
16     RETURN bookingsCursor;
17 END get_customer_bookings;
```

הסבר get_order

הפונקציה `get_order` היא פונקציה ב-PL/SQL שמחזירה רשימה של מזהי הזמנות בהתבסס על מזהה לקוח נתון. הנה הסבר קצר על מה שהיא עושה:

הפונקציה מקבלת פרמטר אחד בשם `p_c_id` שהוא מספר (NUMBER).

הפונקציה מחזירה אובייקט מסוג `SYS_REFCURSOR`, שהוא מצביע לרשומות שנבחרו.

משתנה מקומי בשם `orderList` מוגדר כאובייקט `SYS_REFCURSOR`.

בתוך הבלוק של `BEGIN`:

הפונקציה פותחת את הקורסור `orderList` ומבצעת שאילתת SQL שמחזירה את מזהי ההזמנות (`orderID`) מתוך הטבלאות `eventID`, `orderID` ו-`Locations`, בהן קיים חיבור (join) בין הטבלאות על פי תנאים מסוימים.

התנאי שבשאילתה הוא שהעמודה `c_id` בטבלה לא מזוהה להיות שווה לפרמטר `p_c_id`.

הפונקציה סוגרת את הקורסור `bookingsCursor`.

הפונקציה מחזירה את הקורסור `bookingsCursor`.

Discount- prc

Declaration	Variable
1	<code>create or replace procedure Discount(order_id_1 NUMBER ,totalp NUMBER) is</code>
2	<code>TotalPcont NUMBER;</code>
3	<code>begin</code>
4	<code>if totalp>250 then</code>
5	<code> TotalPcont:=totalp*0.9;</code>
6	<code>else</code>
7	<code> TotalPcont:=totalp;</code>
8	<code>end if;</code>
9	<code>DBMS_OUTPUT.PUT_LINE('The discounted order cost for order ID ' order_id_1 ' is ' TotalPcont);</code>
10	<code>end Discount;</code>

הסבר prcDiscount-

הפרוצדורה **Discount** היא פרוצדורה ב-PL/SQL שמחשבת ומדפיסה את העלות המוזלת של הזמנה, בהתבסס על סכום כולל של ההזמנה. הנה הסבר קצר על מה שהיא עושה:

הפרוצדורה מקבלת שני פרמטרים:

order_id_1: מספר הזמנה (NUMBER).

totalp: הסכום הכולל של ההזמנה (NUMBER).

משתנה מקומי בשם **TotalPcont** מוגדר לאחסון הסכום הסופי.

בתוך הבלוק של **BEGIN**:

נבדק אם הסכום הכולל של ההזמנה (**totalp**) הוא יותר מ-250.

אם כן, הסכום הסופי (**TotalPcont**) מחושב כ-90% מהסכום הכולל (**totalp * 0.9**).

אם לא, הסכום הסופי נשאר כפי שהוא (**TotalPcont := totalp**).

לבסוף, הפרוצדורה מדפיסה את תוצאת החישוב באמצעות **DBMS_OUTPUT.PUT_LINE**.

בפועל, הפרוצדורה מחשבת הנחה של 10% אם הסכום הכולל של ההזמנה גדול מ-250, ומדפיסה את הסכום המוזל יחד עם מספר ההזמנה.

Test-pr.1

Test script

DBMS Output

Statistics

Profiler

Trace

Clear

Buffer size

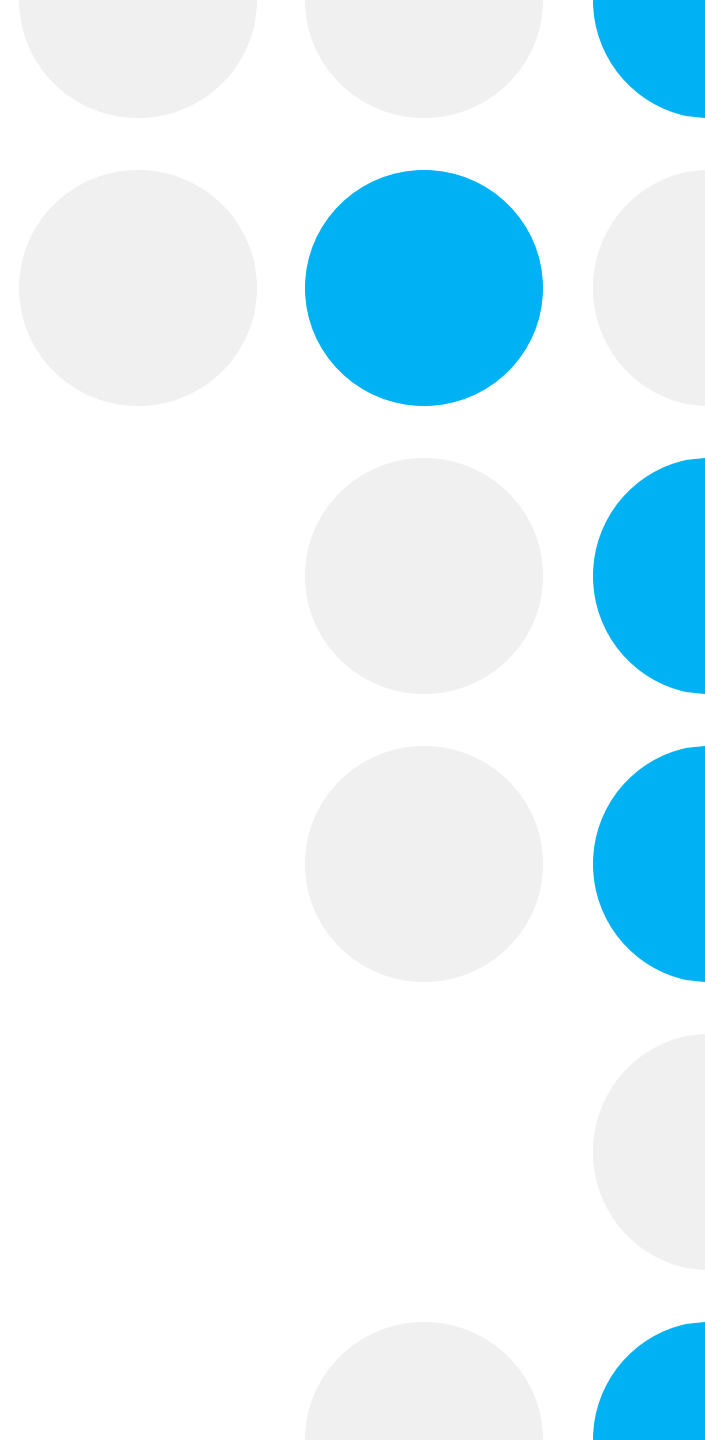
10000

☒ Enabled

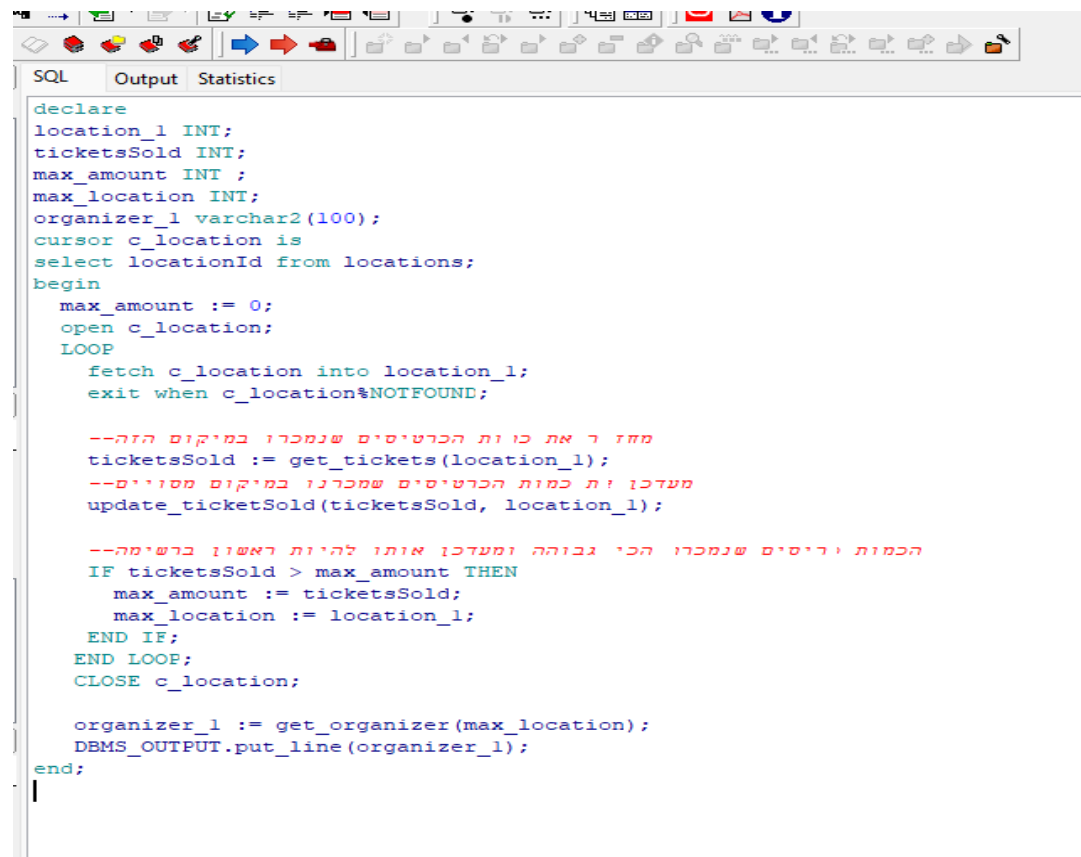
location id 374

order id 175

The discounted order cost for order ID 175 is 10



Program 2



```
SQL Output Statistics
declare
location_1 INT;
ticketsSold INT;
max_amount INT ;
max_location INT;
organizer_1 varchar2(100);
cursor c_location is
select locationId from locations;
begin
    max_amount := 0;
    open c_location;
    LOOP
        fetch c_location into location_1;
        exit when c_location%NOTFOUND;

        --מחזר את כמות הכרטיסים שנמכרו במיקום הזה
        ticketsSold := get_tickets(location_1);
        --מעדכן את כמות הכרטיסים שנמכרו במיקום מסוים
        update_ticketSold(ticketsSold, location_1);

        --הכמות וריסים שנמכרו הכי גבוהה ומעדכן אותו להיות ראשון ברשימה
        IF ticketsSold > max_amount THEN
            max_amount := ticketsSold;
            max_location := location_1;
        END IF;
    END LOOP;
    CLOSE c_location;

    organizer_1 := get_organizer(max_location);
    DBMS_OUTPUT.put_line(organizer_1);
end;
```

Program 2

התוכנית עושה את הדברים הבאים:

מגדירה משתנים וקורסור לשליפת מזהי מיקום.

עוברת על כל המיקומים בטבלה `locations` באמצעות הקורסור.

לכל מיקום:

שולפת את מספר הכרטיסים שנמכרו בעזרת `.get_tickets`.

מעדכנת את מספר הכרטיסים שנמכרו בעזרת `.update_ticketSold`.

בודקת אם זהו המספר המרבי שנמכר עד כה ומעדכנת את המשתנים `max_amount` -

`max_location` בהתאם.

סוגרת את הקורסור.

שולפת את שם המארגן במיקום עם המספר המרבי של כרטיסים שנמכרו בעזרת `.get_organizer`.

מדפיסה את שם המארגן.

get_organizer-fnc

Declaration		Variable
1	<input type="checkbox"/>	create or replace function get_organizer(location INT) return varchar2 is
2	<input type="checkbox"/>	organizerNmae organizer.organizername%type;
3		phone_o organizer.phone%type;
4		email_o organizer.email%type;
5		begin
6	<input type="checkbox"/>	select o.organizername, o.phone, o.email
7		INTO organizerNmae, phone_o, email_o
8		from organizer o JOIN event e on o.organizerid = e.organizerid JOIN locations l ON l.locationid = e.locationid
9		where l.locationid = location
10		and rownum = 1;
11		return 'organizer name- ' organizerNmae ' phone- ' phone_o ' email- ' email_o;
12		EXCEPTION
13	<input type="checkbox"/>	WHEN NO_DATA_FOUND THEN
14		RETURN NULL;
15	<input type="checkbox"/>	when others
16		then raise;
17		
18		end get_organizer;

get_organizer-fnc

הפונקציה `get_organizer` ב-PL/SQL מקבלת מזהה מיקום (`location`) ומחזירה מחרוזת עם פרטי המארגן (שם, טלפון, ודוא"ל) במיקום הנתון. הנה הסבר קצר על מה שהיא עושה:

הפונקציה מקבלת פרמטר `location` מסוג `INT` ומחזירה מחרוזת (`VARCHAR2`).

מגדירה משתנים לאחסון שם המארגן, טלפון ודוא"ל.

בתוך הבלוק `BEGIN`:

מבצעת שאילתת SQL שמשלבת את הטבלאות `event`, `organizer`, ו-`locations` כדי לשלוף את שם המארגן, טלפון ודוא"ל עבור המיקום הנתון.

התוצאות נשמרות במשתנים המקומיים.

מחזירה מחרוזת שמכילה את שם המארגן, טלפון ודוא"ל.

טיפול בשגיאות:

אם לא נמצאו נתונים (`NO_DATA_FOUND`), מחזירה `NULL`.

במקרה של שגיאה אחרת, מעלה מחדש את השגיאה.

בפועל, הפונקציה שולפת את פרטי המארגן עבור מיקום נתון ומחזירה אותם כמחרוזת אחת.

get_tickets-fnc

```
1 create or replace function get_tickets(location_l INT) return INT is
2   ticket_sold INT ;
3   begin
4     select count(o.orderid)
5     into ticket_sold
6     from Orders o JOIN event e ON o.eventId = e.eventid
7     WHERE e.locationid = location_l;
8
9     return(ticket_sold);
10  end get_tickets;
```


get_tickets-fnc

הפונקציה `get_tickets` ב-PL/SQL מקבלת מזהה מיקום (`location_1`) ומחזירה את מספר הכרטיסים שנמכרו עבור אותו מיקום. הנה הסבר קצר על מה שהיא עושה:

הפונקציה מקבלת פרמטר `location_1` מסוג `INT` ומחזירה מספר שלם (`INT`).

מגדירה משתנה `ticket_sold` לאחסון מספר הכרטיסים שנמכרו.

בתוך הבלוק `BEGIN`:

מבצעת שאילתת SQL שמשלבת את הטבלאות `Orders` ו-`event` כדי לספור את מספר ההזמנות (`orderid`) עבור מיקום נתון.

התוצאה נשמרת במשתנה `ticket_sold`.

מחזירה את ערך `ticket_sold`.

בפועל, הפונקציה מחשבת ומחזירה את מספר הכרטיסים שנמכרו עבור מיקום מסוים.

update_ticketSold-prc

```
1 create or replace procedure update_ticketSold(ticketSold1 INT, loction_id INT) is
2
3 begin
4     update locations
5     set ticketsSold = ticketSold1;
6     where locationId = loction_id;
7
8     DBMS_OUTPUT.put_line(loction_id || '-' || ticketSold1);
9
10 end update_ticketSold;
```

-The end-

