

הצגת כל אירועים באירועים האחרונים - AllEventsXWeeks

מטרתה - להציג את כל האירועים שהתרחשו ב-X-השבועות האחרונים, כאשר X הוא פרמטר שנקבע על ידי המשתמש. בנוסף, השאלה מציגה את מחיר האירוע של כל אירוע.

השאילתת מתחילה על ידי בחירת כל השדות מטבלת האירועים (dbCourseSt23.team_21_Event) ומחייב האירוע מטבלת הזמינות (dbCourseSt23.team_21_Order). היא מחברת את שתי הטבלאות על ידי שימוש בJOIN-כasher התנאי לחיבור הוא שמהזאה של אירוע (id) בטבלת האירועים יהיה שווה למזהה של אירוע (EventID) בטבלת הזמינות.

לאחר מכן, השאלה מסננת את התוצאות כך שרק אירועים שהתרחשו ב-X-השבועות האחרונים יוצגו. זה מתבצע על ידי שימוש בפונקציה DATE_SUB(
DATE_CURDATE()) שמחזירה את התאריך הנוכחי לפני התאריך הנוכחי. השאילתת מחזירה את כל האירועים שתאריך האירוע שלהם (EventDate) הוא בין התאריך שהוחזר מהפונקציה DATE_SUB(
CURDATE()) והתאריך הנוכחי.

בקוד ה-PHP, השאלה מוכנה על ידי שימוש בפונקציה prepare של אובייקט החיבור למסד הנתונים (\$conn). הפרמטר X מועבר לשאילתת על ידי שימוש בפונקציה bind_param לבסוף, השאלה מוצאת לפועל על ידי שימוש בפונקציה execute.

השאילתת:

```
SELECT e.*, o.eventPrice  
  
FROM dbCourseSt23.team_21_Event AS e  
  
JOIN dbCourseSt23.team_21_Order AS o ON e.id = o.EventID  
  
WHERE e.EventDate BETWEEN DATE_SUB(CURDATE(), INTERVAL ? WEEK) AND CURDATE();
```

-- PHP --

```
$stmt = $conn->prepare("SELECT e.* , o.eventPrice  
FROM dbCourseSt23.team_21_Event AS e  
JOIN dbCourseSt23.team_21_Order AS o ON e.id = o.EventID  
WHERE e.EventDate BETWEEN DATE_SUB(CURDATE(), INTERVAL ? WEEK) AND  
CURDATE());  
  
$stmt->bind_param("i", $weeks);
```

הציגת האירועים הפעילים (שהairoうם שלהם עוד לא התקיימ) והלקוח שהזמין -

AllEventsCustomerOrdered

השאילתה מציגה את האירועים הפעילים, כוללם את האירועים שהתאריך שלהם עוד לא התקיים. היא מציגה גם את הלקוח שהזמין את האירוע.

השאילתה מתחילה בבחירה מספר שדות מטבלת ההזמנות (Order), הלקוחות (Customer), והאירועים (Event). היא מציגה את מזזה ההזמנה, שם הלקוח, שם המוכר שניהל את הזמנה, סוג/שם האירוע, מחיר האירוע, תאריך האירוע, מספר האורחים, ומהירות המינימלי לאירוע.

השאילתה לחברת את טבלת ההזמנות (Order) עם טבלת האירועים (Event) על ידי שימוש בmezhaה האירוע, ואת טבלת ההזמנות עם טבלת הלקוחות (Customer) על ידי שימוש בmezhaה הלקוח.

לבסוף, השאילתה מסננת את התוצאות כך שרק האירועים שהתאריך שלהם גדול מהתאריך הנוכחי (CURDATE()) מוצגים. זה מבטיח שרק האירועים הפעילים מוצגים.

בקוד ה-PHP, השאילתה מוצאת לפועל על ידי שימוש בפונקציה query של אובייקט החיבור למסד הנתונים (\$conn). התוצאות מוחזרות כאובייקט שניtin לעבור עליו באמצעות לולה כדי להציג את הנתונים.

השאילתה:

```
SELECT
    o.id,
    c.name AS CustomerName,
    (SELECT name FROM dbCourseSt23.team_21_Employee WHERE id = o.salespersonId) AS
    SalespersonName,
    e.eventType,
    o.eventPrice,
    e.eventDate,
    e.numberOfGuests,
    e.minimumPrice
FROM
    dbCourseSt23.team_21_Order AS o
JOIN
    dbCourseSt23.team_21_Event AS e ON o.EventID = e.id
JOIN
    dbCourseSt23.team_21_Customer AS c ON o.CustomerID = c.id
WHERE
    e.EventDate > CURDATE();
```

AllEventsMissingEmployee - הצגת אירועים שהסרים להם מלצרים וטבחים

השאילתת מציגה את האירועים שהסרים להם מלצרים או טבחים. היא מציגה את מזגה האירוע, תאריך האירוע, מספר המלצרים הנדרשים, מספר הטבחים הנדרשים, מספר המלצרים שהוקצו כבר לאירוע, מספר הטבחים שהוקצו כבר לאירוע, מספר המלצרים החסרים, ומספר הטבחים החסרים.

השאילתת משתמשת בשאלות משנה כדי לחשב את מספר המלצרים והטבחים שהוקצו לכל אירוע. השאלות משנה מחבירות את טבלת האירועים (Event) עם טבלת העובדים (Employee) דרך טבלת הקישור EventEmployee, ומסנוות את התוצאות לפי התפקיד (מלצר או טבח).

לאחר מכן, השאילתת מחשבת את מספר המלצרים והטבחים החסרים על ידי הפקחת מספר העובדים שהוקצו מהמספר הנדרש.

בסוף, השאילתת מסננת את התוצאות כך שרק האירועים שהסרים להם מלצרים או טבחים מוצגים. זה מבטיח שרק אירועים שיש להם צורך בעובדים נוספים מוצגים.

השאילתת:

```
SELECT
    e.id,
    e.eventType,
    e.eventDate,
    e.numberOfWaitersRequired,
    e.numberOfCooksRequired,
    (SELECT COUNT(*) FROM dbCourseSt23.team_21_EventEmployee AS ee
     JOIN dbCourseSt23.team_21_Employee AS emp ON ee.employeeId = emp.id
     WHERE ee.eventId = e.id AND emp.role = 'Waiter') AS AssignedWaiters,
    (SELECT COUNT(*) FROM dbCourseSt23.team_21_EventEmployee AS ee
     JOIN dbCourseSt23.team_21_Employee AS emp ON ee.employeeId = emp.id
     WHERE ee.eventId = e.id AND emp.role = 'Cook') AS AssignedCooks,
    e.numberOfWaitersRequired - (SELECT COUNT(*) FROM dbCourseSt23.team_21_EventEmployee AS ee
     JOIN dbCourseSt23.team_21_Employee AS emp ON ee.employeeId = emp.id
     WHERE ee.eventId = e.id AND emp.role = 'Waiter') AS WaitersMissing,
    e.numberOfCooksRequired - (SELECT COUNT(*) FROM dbCourseSt23.team_21_EventEmployee AS ee
     JOIN dbCourseSt23.team_21_Employee AS emp ON ee.employeeId = emp.id
     WHERE ee.eventId = e.id AND emp.role = 'Cook') AS CooksMissing
FROM
    dbCourseSt23.team_21_Event AS e
HAVING
    WaitersMissing > 0 OR CooksMissing > 0
```

ClinetsMoreThanOneOrder – הציגת לקוחות חוזרים

השאילתה מציגה את לקוחות שביצעו יותר מפעם אחת. היא מציגה את מזהה הלוקח, שם הלוקח, ומספר ההזמנות שביצע הלוקח.

השאילתה מחברת את טבלת הלקוחות (Customer) עם טבלת ההזמנות (Order) על פי מזהה הלוקח. היא מקובצת את התוצאות לפי מזהה הלוקח ושם הלוקח, ומחשבת את מספר ההזמנות שביצע כל לקוח.

לאחר מכן, השאילתה מסננת את התוצאות כך שרק לקוחות שביצעו יותר מפעם אחת מוצגים.

השאילתה:

```
SELECT c.id, c.name, COUNT(o.id) AS NumberOfOrders  
FROM dbCourseSt23.team_21_Customer AS c  
JOIN dbCourseSt23.team_21_Order AS o ON c.id = o.customerId  
GROUP BY c.id, c.name  
HAVING COUNT(o.id) > 1;
```

הצגת הכנסות ב-X החודשים האחרונים - AllIncomeXMonths

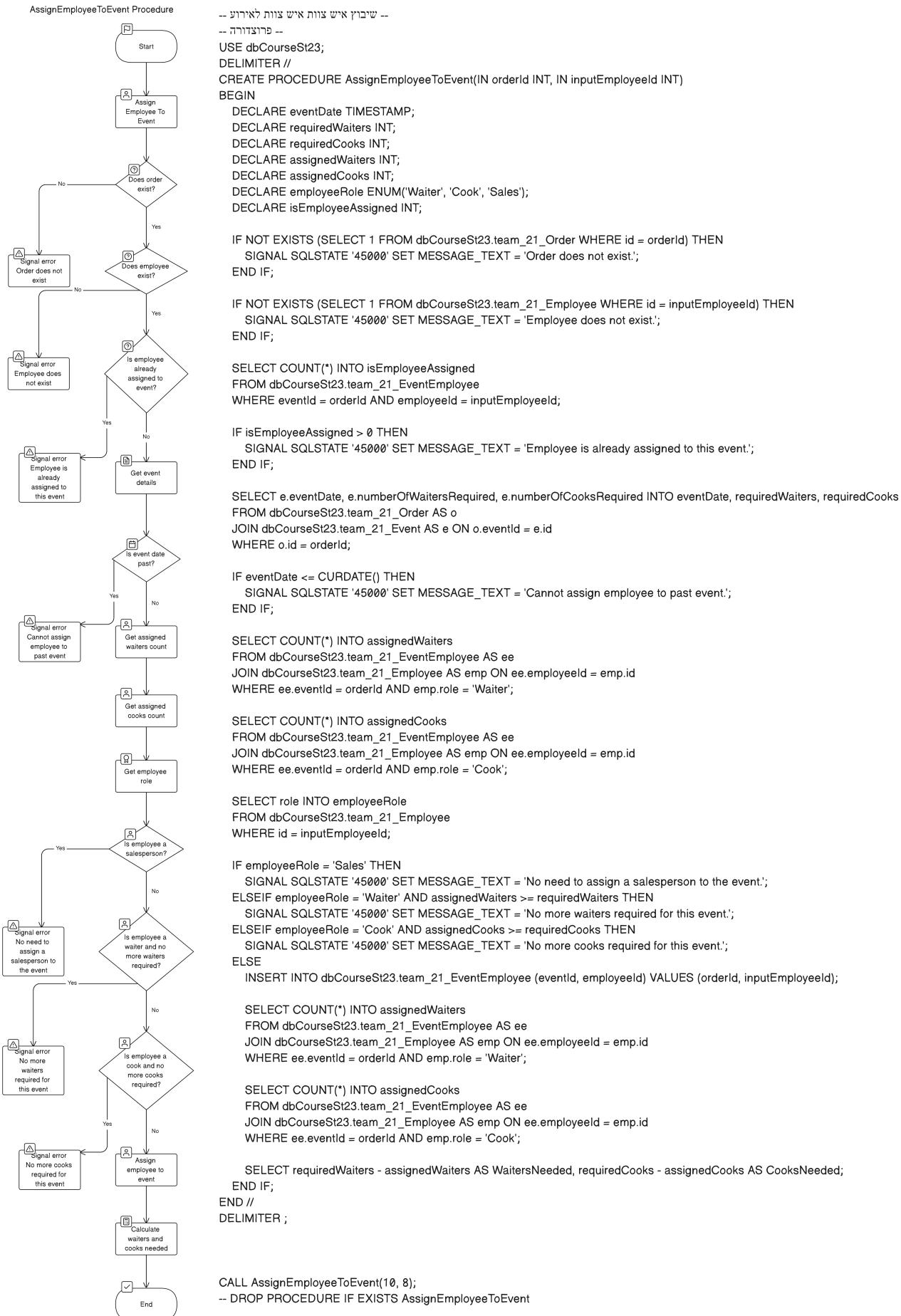
השאילתה מציגה את סך הכנסות מהאירועים שהתרחשו ב-X-החודשים האחרונים, כאשר X הוא פרמטר שמועבר לשאילתת.

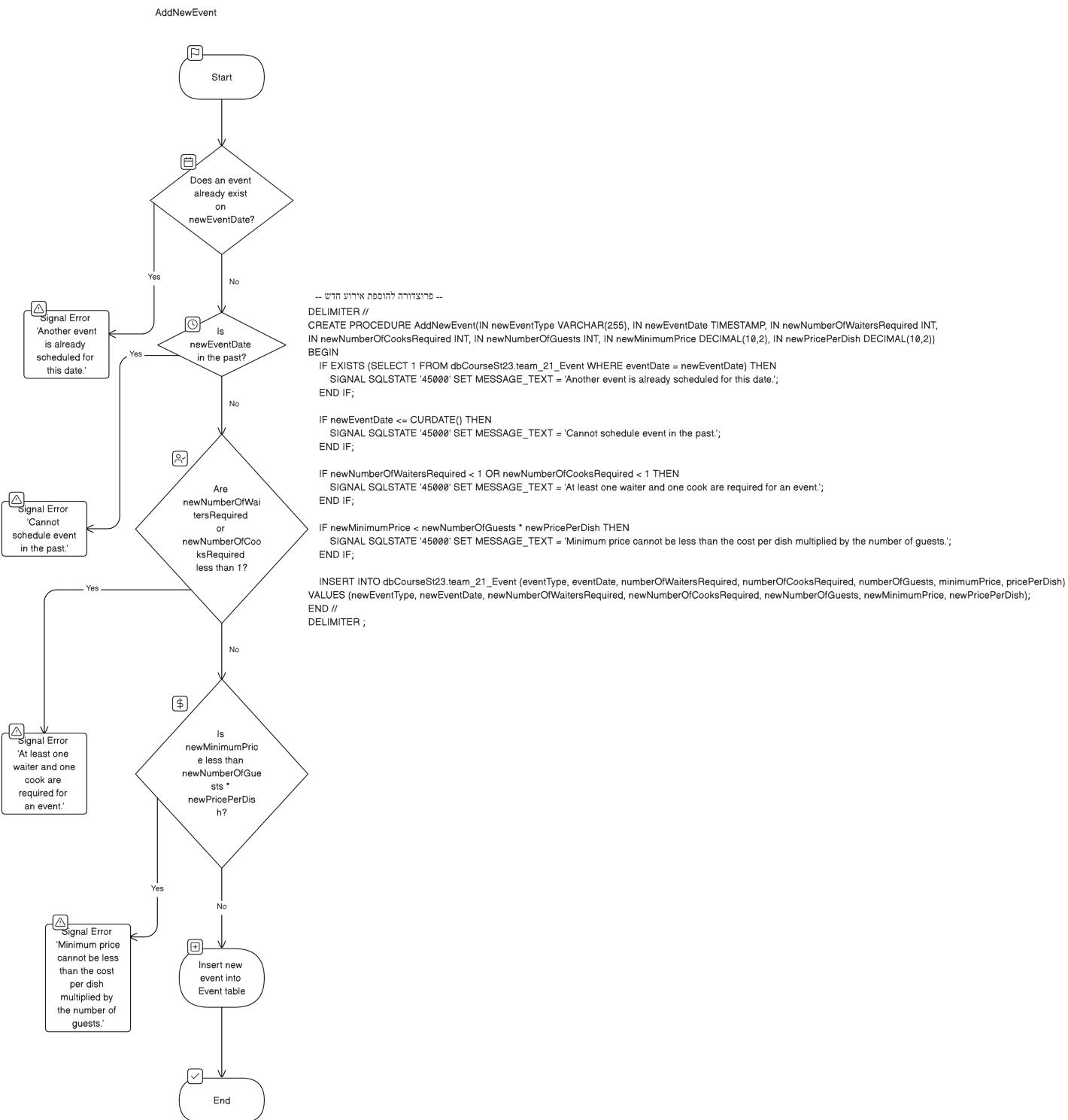
השאילתה מחברת את טבלת ההזמנות (Order) עם טבלת האירועים (Event) על פי מזחה אירוע. היא מסננת את אירועים שהתרחשו בין התאריך הנוכחי פחות X חודשים ועד התאריך הנוכחי.

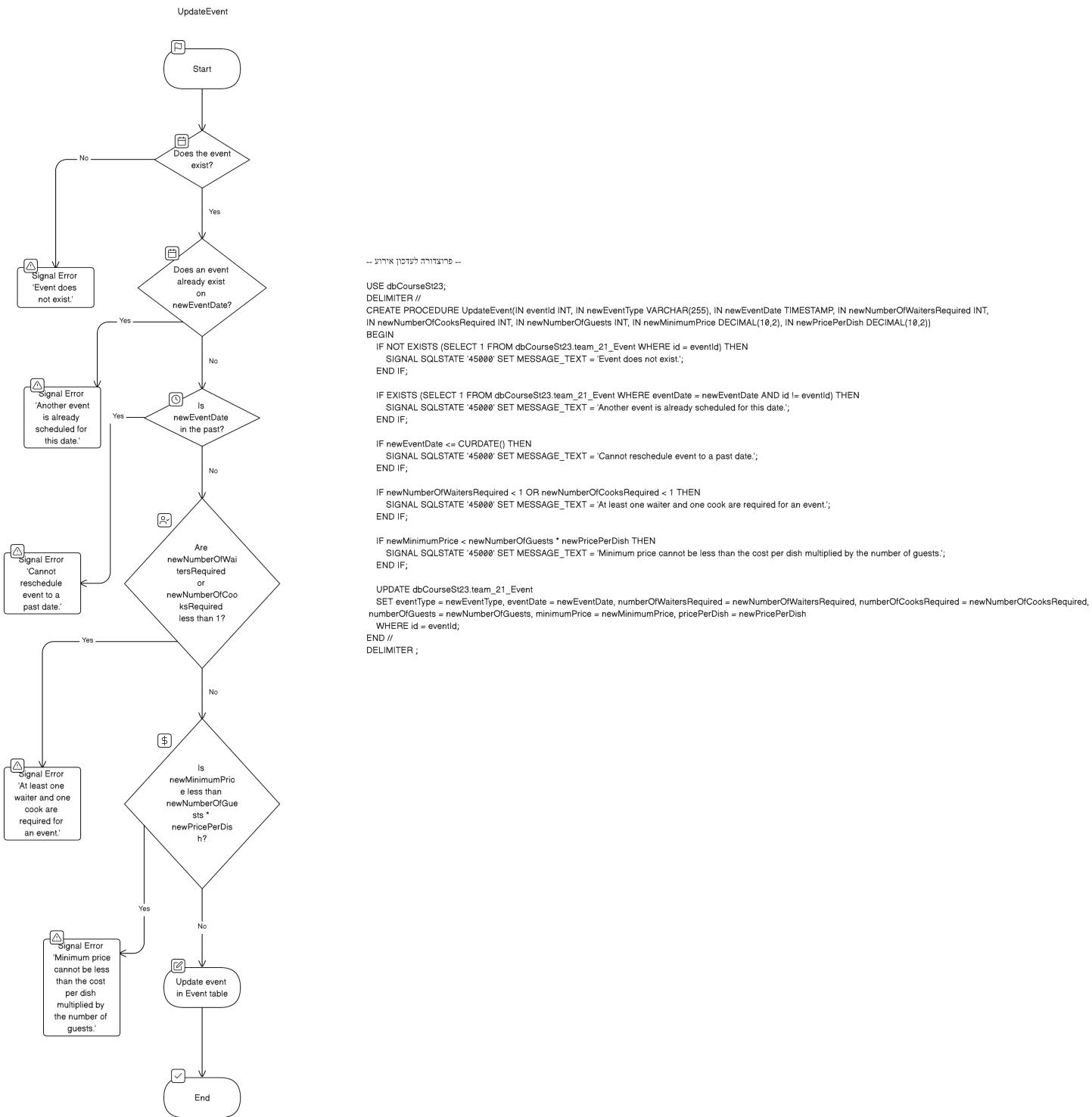
לאחר מכן, השאילתה מחשבת את סך מחיר אירוע של כל ההזמנות שנבחרו, ומהזירה את הסכום כ"הכנסה כוללת" (TotalIncome).

השאילתה:

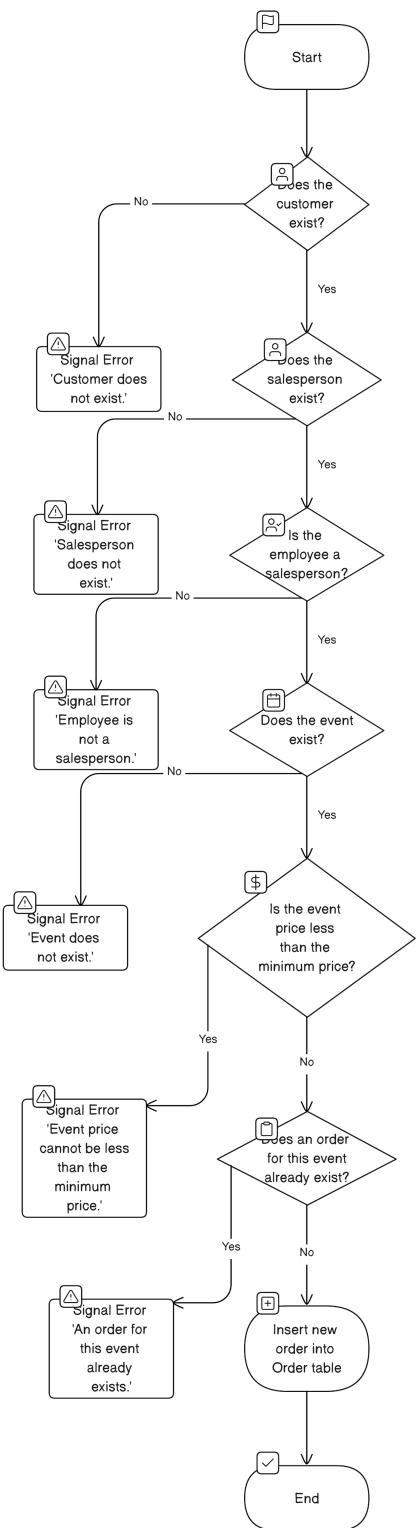
```
SELECT SUM(o.eventPrice) AS TotalIncome FROM dbCourseSt23.team_21_Order AS o
JOIN dbCourseSt23.team_21_Event AS e ON o.eventId = e.id
WHERE e.eventDate BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 MONTH) AND CURDATE();
```







AddNewOrder



-- פונקציית חישוב הינה --

```

USE dbCourseSt23;
DELIMITER //
CREATE PROCEDURE AddNewOrder(IN newCustomerId INT, IN newSalespersonId INT, IN newEventId INT, IN newEventPrice DECIMAL(10, 2))
BEGIN
  DECLARE salespersonRole ENUM('Waiter', 'Cook', 'Sales');

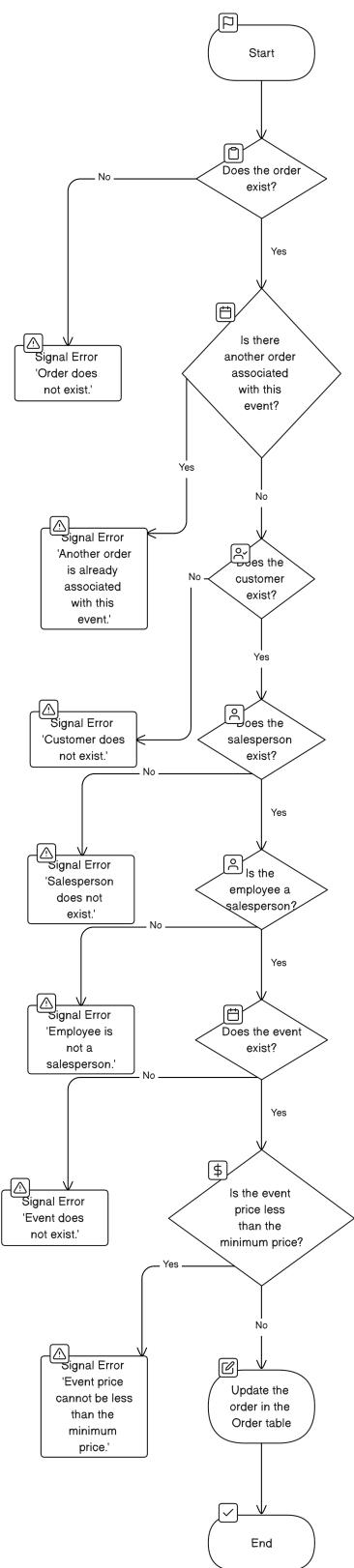
  IF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Customer WHERE id = newCustomerId) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Customer does not exist.';
  ELSEIF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Employee WHERE id = newSalespersonId) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Salesperson does not exist';
  ELSE
    SELECT role INTO salespersonRole
    FROM dbCourseSt23.team_21_Employee
    WHERE id = newSalespersonId;

    IF salespersonRole != 'Sales' THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Employee is not a salesperson.';
    ELSEIF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Event WHERE id = newEventId) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Event does not exist';
    ELSEIF (SELECT minimumPrice FROM dbCourseSt23.team_21_Event WHERE id = newEventId) > newEventPrice THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Event price cannot be less than the minimum price.';

    ELSEIF EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Order WHERE eventId = newEventId) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'An order for this event already exists.';

    ELSE
      INSERT INTO dbCourseSt23.team_21_Order (customerId, salespersonId, eventId, eventPrice)
      VALUES (newCustomerId, newSalespersonId, newEventId, newEventPrice);
    END IF;
  END IF;
END //
DELIMITER ;
  
```

UpdateOrder



```

-- עדכון ההזמנה --
USE dbCourseSt23;
DELIMITER //
CREATE PROCEDURE UpdateOrder(IN orderId INT, IN newCustomerId INT, IN newSalespersonId INT, IN newEventId INT, IN newEventPrice DECIMAL(10, 2))
BEGIN
  DECLARE salespersonRole ENUM('Waiter', 'Cook', 'Sales');

  IF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Order WHERE id = orderId) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Order does not exist.';
  END IF;

  IF EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Order WHERE eventId = newEventId AND id != orderId) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Another order is already associated with this event.';
  END IF;

  IF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Customer WHERE id = newCustomerId) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Customer does not exist.';
  END IF;

  IF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Employee WHERE id = newSalespersonId) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Salesperson does not exist.';
  END IF;

  SELECT role INTO salespersonRole
  FROM dbCourseSt23.team_21_Employee
  WHERE id = newSalespersonId;

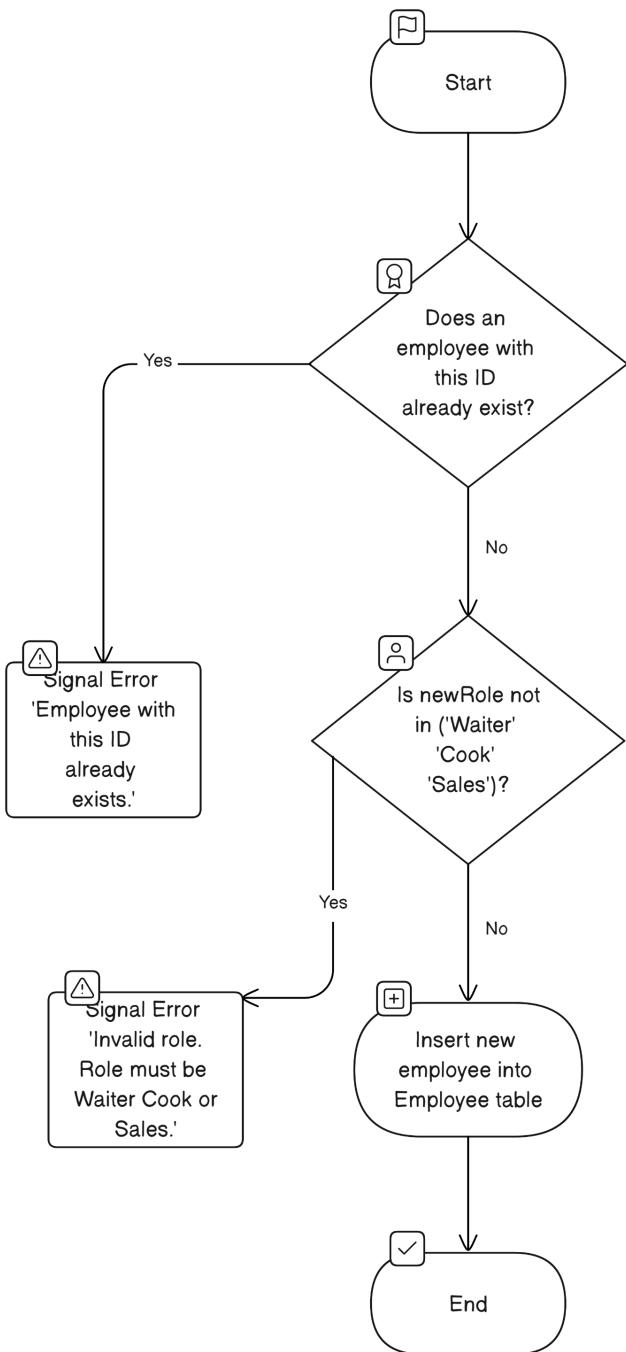
  IF salespersonRole != 'Sales' THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Employee is not a salesperson.';
  END IF;

  IF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Event WHERE id = newEventId) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Event does not exist.';
  END IF;

  IF (SELECT minimumPrice FROM dbCourseSt23.team_21_Event WHERE id = newEventId) > newEventPrice THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Event price cannot be less than the minimum price.';
  END IF;

  UPDATE dbCourseSt23.team_21_Order
  SET customerId = newCustomerId, salespersonId = newSalespersonId, eventId = newEventId, eventPrice = newEventPrice
  WHERE id = orderId;
END //
DELIMITER ;
  
```

AddNewEmployee



-- פרוצדורה הוספה עובד חדש --

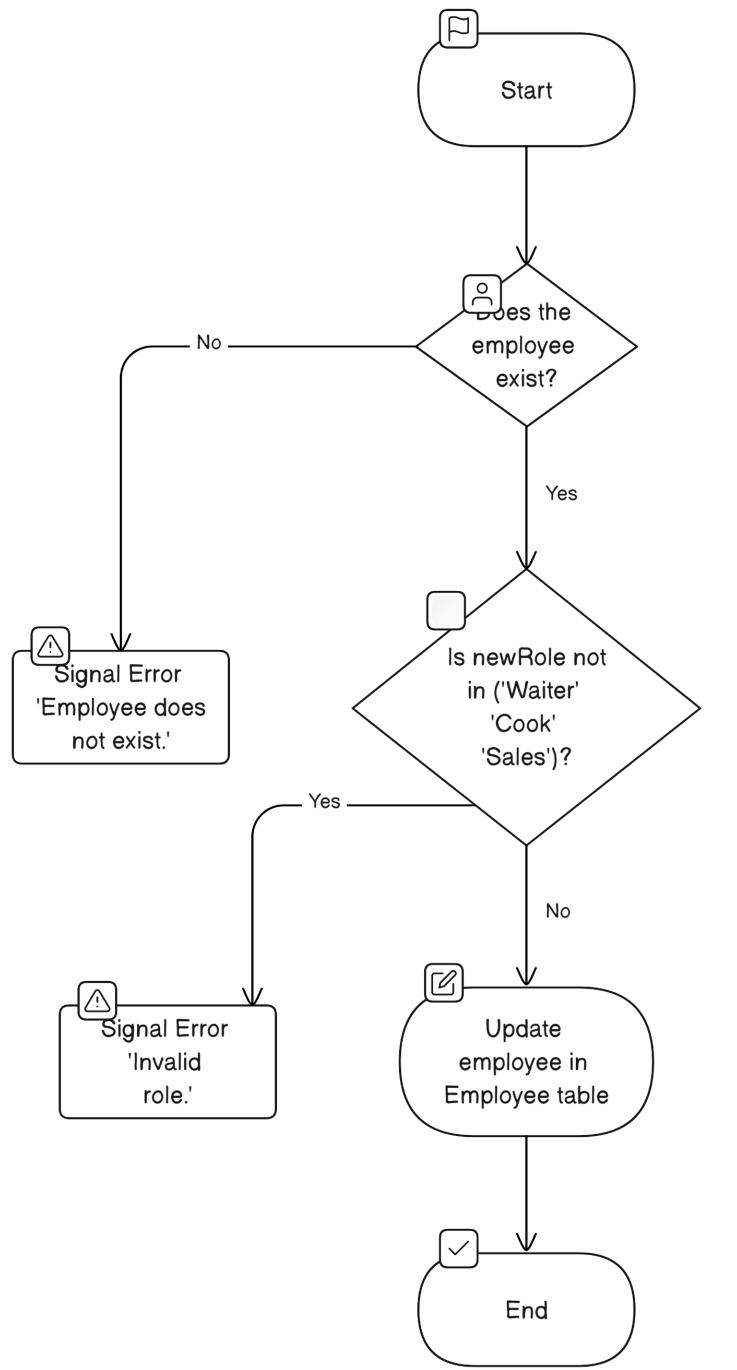
```

USE dbCourseSt23;
DELIMITER //
CREATE PROCEDURE AddNewEmployee(IN newId INT, IN newRole ENUM('Waiter', 'Cook', 'Sales'),
IN newName VARCHAR(255), IN newAddress VARCHAR(255), IN newPhone VARCHAR(255))
BEGIN
  IF EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Employee WHERE id = newId) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Employee with this ID already exists。';
  END IF;

  IF newRole NOT IN ('Waiter', 'Cook', 'Sales') THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid role. Role must be Waiter, Cook, or Sales。';
  END IF;

  INSERT INTO dbCourseSt23.team_21_Employee (id, role, name, address, phone)
VALUES (newId, newRole, newName, newAddress, newPhone);
END //
DELIMITER ;
  
```

UpdateEmployee



-- פרוצדורה עדכון עובד --

```

USE dbCourseSt23;
DELIMITER //
CREATE PROCEDURE UpdateEmployee(IN newId INT, IN newRole ENUM('Waiter', 'Cook', 'Sales'),
IN newName VARCHAR(255), IN newAddress VARCHAR(255), IN newPhone VARCHAR(255))
BEGIN
  IF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Employee WHERE id = newId) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Employee does not exist.';
  END IF;

  IF newRole NOT IN ('Waiter', 'Cook', 'Sales') THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid role.';
  END IF;

  UPDATE dbCourseSt23.team_21_Employee
  SET role = newRole, name = newName, address = newAddress, phone = newPhone
  WHERE id = newId;
END //
DELIMITER ;
  
```

מתן הנחה באחוזים - GiveDiscount

בפרוצדורה "GiveDiscount", המערכת מבצעת את הפעולות הבאות:

המערכת מקבלת שני פרמטרים: inputEventId (מזהה האירוע שלו אנחנו רוצים לשנות את המחיר) ו- discountPercentage (אחוז הנחה שאנו רוצים לתת).

המערכת בודקת אם קיימ אירוע עם המזהה inputEventId. אם לא, היא מחייבת שגיאת שאומרת "האירוע לא קיים".

המערכת בודקת אם discountPercentage הוא 0 או פחות. אם כן, היא מחייבת שגיאת שאומרת "אחוז הנחה חייב להיות גדול מ-0".

המערכת מחפשת את המחיר הנוכחי של האירוע ואת המחיר המינימלי שלו.

המערכת מחשבת את המחיר החדש של האירוע לאחר הנחה.

המערכת בודקת אם המחיר החדש של האירוע הוא פחות מהמחיר המינימלי שלו. אם כן, היא מחייבת שגיאת שאומרת "המחיר לאחר הנחה לא יכול להיות נמוך מהמחיר המינימלי".

אם כל הבדיקות עברו בהצלחה, המערכת מעדכנת את המחיר של האירוע בטבלת ההזמנות.

הפרוצדורה:

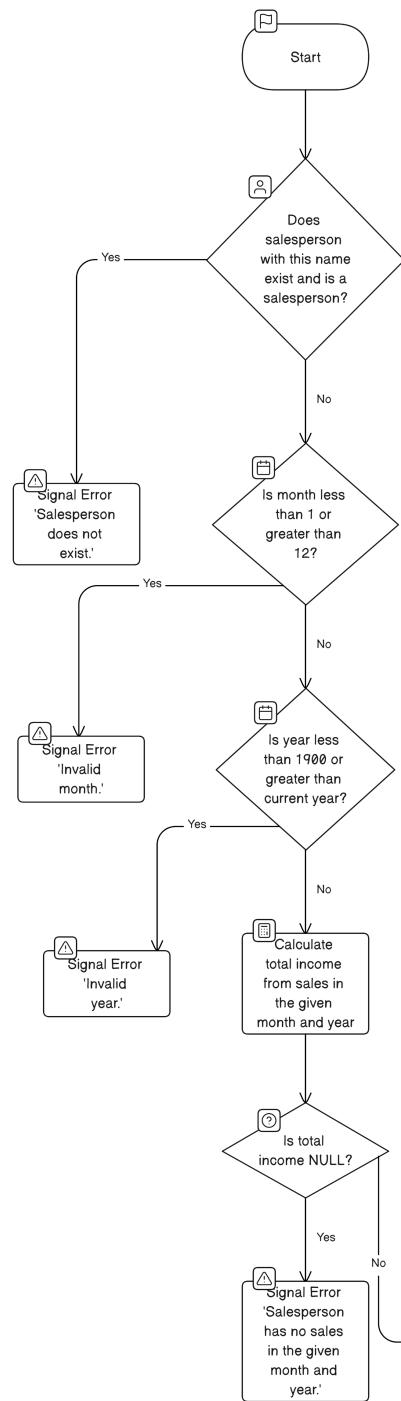
```
-- מתן הנחה באחוזים --
DELIMITER //
CREATE PROCEDURE GiveDiscount(IN inputEventId INT, IN discountPercentage DECIMAL(5, 2))
BEGIN
    DECLARE currentPrice DECIMAL(10, 2);
    DECLARE newPrice DECIMAL(10, 2);
    DECLARE minimumPrice DECIMAL(10, 2);

    IF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Event WHERE id = inputEventId) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Event does not exist.';
    END IF;
    IF discountPercentage <= 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Discount percentage must be greater than 0.';

    END IF;
    SELECT o.eventPrice, e.minimumPrice INTO currentPrice, minimumPrice
    FROM dbCourseSt23.team_21_Order o
    JOIN dbCourseSt23.team_21_Event e ON o.eventId = e.id
    WHERE o.eventId = inputEventId;
    SET newPrice = currentPrice - (currentPrice * discountPercentage / 100);
    IF newPrice < minimumPrice THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Discounted price cannot be less than the
minimum price.';

    END IF;
    UPDATE dbCourseSt23.team_21_Order
    SET eventPrice = newPrice
    WHERE eventId = inputEventId;
END //
DELIMITER ;
```

CalculateSalespersonIncome_function



לכל איש מכירות את סכום ההכנסות לחודש מסוים

```

USE dbCourseSt23;
DELIMITER //
CREATE FUNCTION CalculateSalespersonIncome(salespersonName VARCHAR(255), month INT, year INT) RETURNS DECIMAL(10,2)
BEGIN
  DECLARE totalIncome DECIMAL(10,2);

  IF NOT EXISTS (SELECT 1 FROM dbCourseSt23.team_21_Employee WHERE name = salespersonName AND role = 'Sales') THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Salesperson does not exist.';
  END IF;

  IF month < 1 OR month > 12 THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid month.';
  END IF;

  IF year < 1900 OR year > YEAR(CURDATE()) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid year.';
  END IF;

  SELECT SUM(o.eventPrice)
  INTO totalIncome
  FROM dbCourseSt23.team_21_Order o
  JOIN dbCourseSt23.team_21_Employee e ON o.salespersonId = e.id
  JOIN dbCourseSt23.team_21_Event ev ON o.eventId = ev.id
  WHERE e.name = salespersonName AND MONTH(ev.eventDate) = month AND YEAR(ev.eventDate) = year;

  IF totalIncome IS NULL THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Salesperson has no sales in the given month and year.';
  END IF;

  RETURN totalIncome;
END //
DELIMITER ;
  
```