דו"ח שלב 4:  
  
  
  
**פונקציה 1: סיכום הכנסות ותשלומים לפי שנה**  
  
CREATE OR REPLACE FUNCTION get\_yearly\_revenue\_summary(p\_year INTEGER)

RETURNS TABLE (

year INTEGER,

total\_revenue NUMERIC(10,2),

num\_payments INTEGER

) AS

$$

DECLARE

v\_total\_revenue NUMERIC(10,2);

v\_num\_payments INTEGER;

BEGIN

-- קבלת סך ההכנסות לשנה המבוקשת

SELECT COALESCE(SUM(tr.total\_revenue), 0)

INTO v\_total\_revenue

FROM taxreport tr

WHERE tr.tax\_year = p\_year;

-- קבלת מספר התשלומים לשנה המבוקשת

SELECT COUNT(\*) INTO v\_num\_payments

FROM payment P

JOIN revenue R ON P.revenue\_id = R.revenue\_id

WHERE R.year = p\_year;

-- אם אין הכנסות ואין תשלומים – נזרוק שגיאה

IF v\_total\_revenue = 0 AND v\_num\_payments = 0 THEN

RAISE EXCEPTION 'לא נמצאו נתונים לשנה %', p\_year;

END IF;

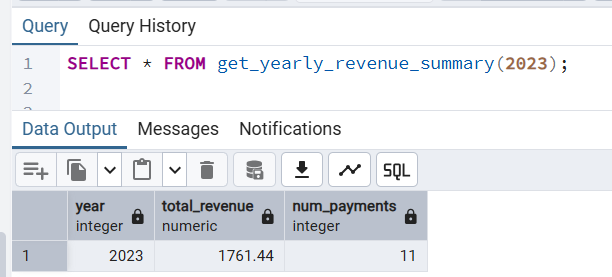
-- החזרת השורה

RETURN QUERY SELECT p\_year, v\_total\_revenue, v\_num\_payments;

END;

$$ LANGUAGE plpgsql;  
יצירת הפונקציה:



הרצת הפונקציה:  
  


\*\*שמנו לב שהנתונים שמוכנסים לא כל כך מעודכנים ולכן:  
לכל שורת payment, נרצה לעדכן את payment\_date כך שתהיה **תוך אותה שנה** שאליה שייכת ההכנסה (revenue.year),   
עדכון השנה בלבד בטבלת payment:  
  
UPDATE payment p

SET payment\_date = make\_date(

r.year,

EXTRACT(MONTH FROM p.payment\_date)::int,

LEAST(EXTRACT(DAY FROM p.payment\_date)::int, 28) -- לוודא שאין שגיאה בלוח שנה

)

FROM revenue r

WHERE p.revenue\_id = r.revenue\_id;  
  
  
  
  
  
**פרוצדורה 1: עדכון סטטוס חשבוניות ל"שולמה"**  
  
 מחפשת חשבוניות שעדיין אין להן סטטוס (status\_id IS NULL) או שאינן "Paid"

 בודקת אם **בוצע להן תשלום מלא** (התשלום הקשור אליהן שווה לסכום החשבונית)

 אם כן → מעדכנת את status\_id ל־(נניח) 1 (הנחה שזה קוד ל־"Paid")

CREATE OR REPLACE PROCEDURE update\_paid\_invoices()

LANGUAGE plpgsql

AS $$

DECLARE

v\_paid\_id INTEGER;

r\_invoice RECORD;

BEGIN

-- שליפת הקוד של הסטטוס "Paid"

SELECT status\_id INTO v\_paid\_id

FROM invoicestatus

WHERE status\_name = 'Paid';

IF v\_paid\_id IS NULL THEN

RAISE EXCEPTION 'לא קיים סטטוס בשם Paid בטבלת invoicestatus';

END IF;

-- מעבר על כל החשבוניות שעוד לא סומנו כ"שולמה"

FOR r\_invoice IN

SELECT i.invoice\_id, i.amount, p.amount AS paid\_amount

FROM invoice i

JOIN payment p ON i.payment\_id = p.payment\_id

WHERE i.status\_id IS DISTINCT FROM v\_paid\_id

LOOP

-- אם הסכום ששולם זהה לסכום החשבונית – נעדכן ל"Paid"

IF r\_invoice.amount = r\_invoice.paid\_amount THEN

UPDATE invoice

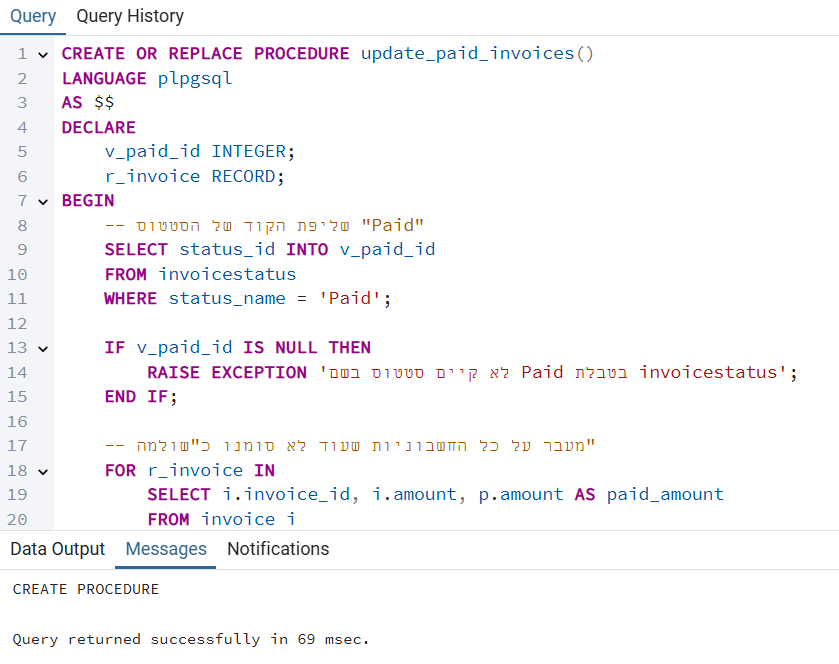
SET status\_id = v\_paid\_id

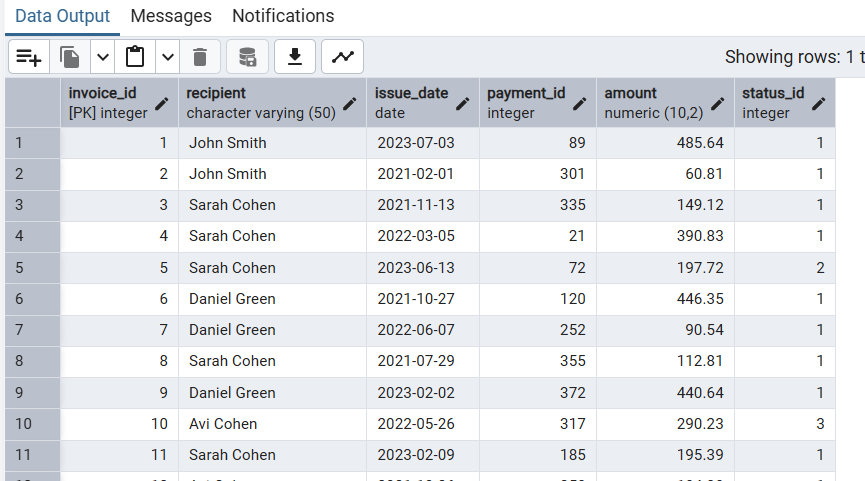
WHERE invoice\_id = r\_invoice.invoice\_id;

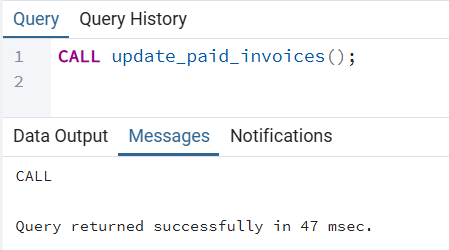
END IF;

END LOOP;

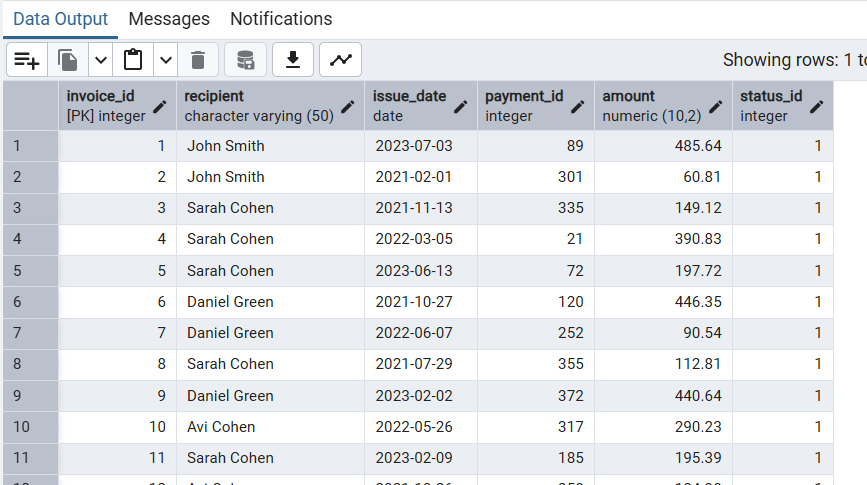
END;

$$;  
  
  
יצירת הפרוצדורה:  
  
  
חלק מנתוני הטבלה לפני הרצת הפרוצדורה:



הרצת הפרוצדורה :  


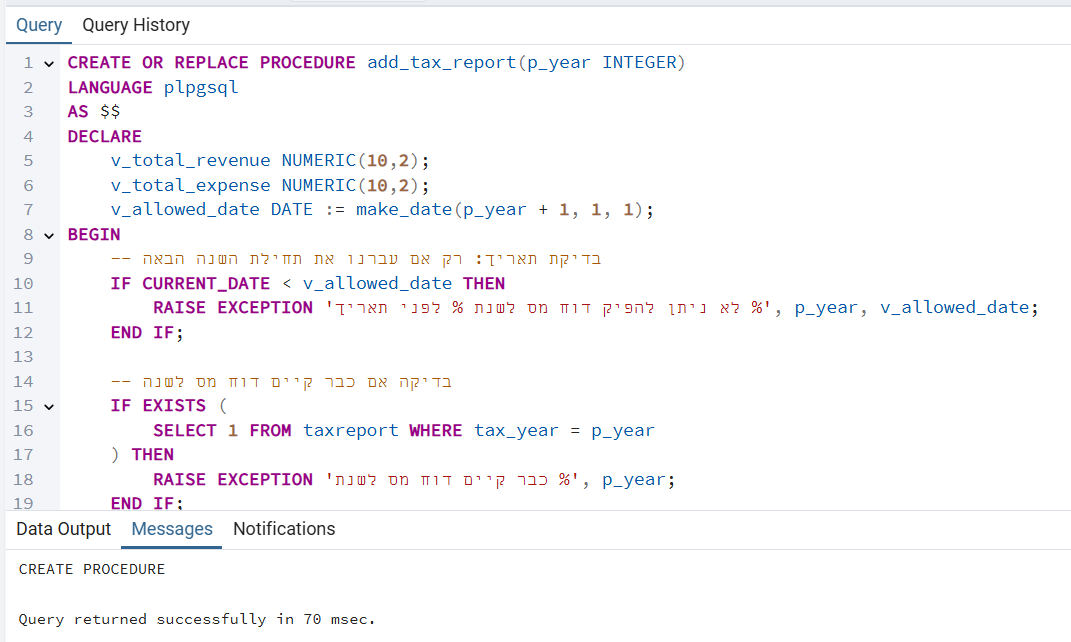
חלק מהטבלה לאחר ההרצה:



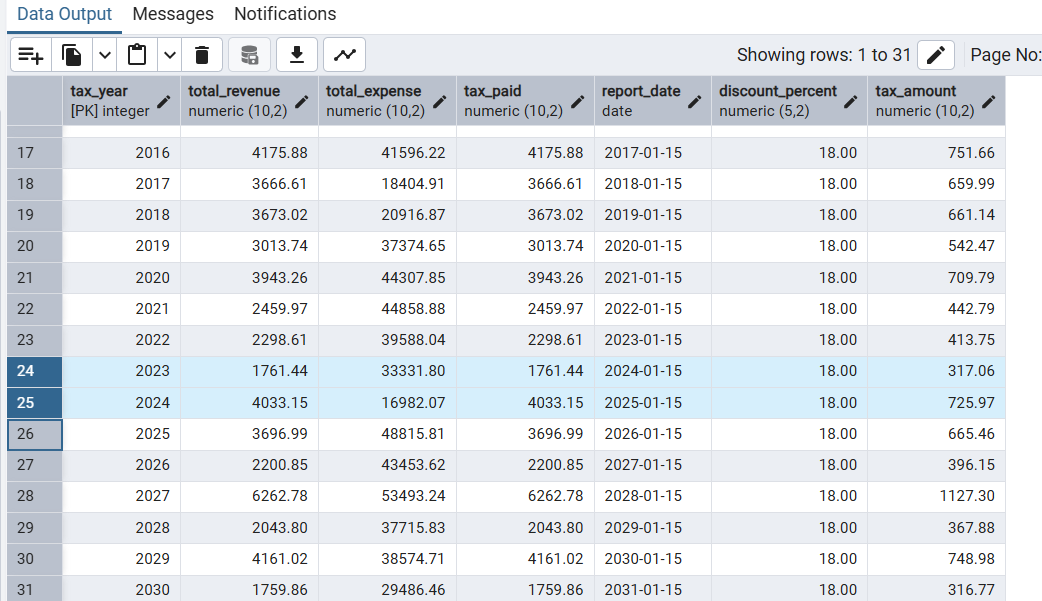
**פרוצדורה 2 –: יצירת רשומת דיווח מס (taxreport) אוטומטית**

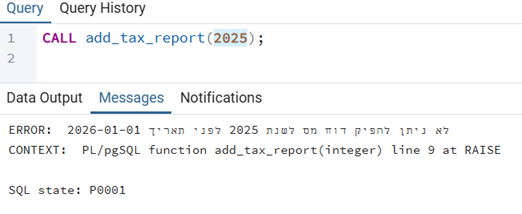
* מקבלת שנה כקלט
* מחפשת את כל ההכנסות וההוצאות מהשנה הזו
* מחשבת:
  + total\_revenue
  + total\_expense
* יוצרת רשומה חדשה בטבלת taxreport
* אבל היא **תתבצע רק אם התאריך הנוכחי הוא מינימום 1 בינואר של השנה הבאה**
* אם מפעילים אותה מוקדם מדי → תזרוק חריגה

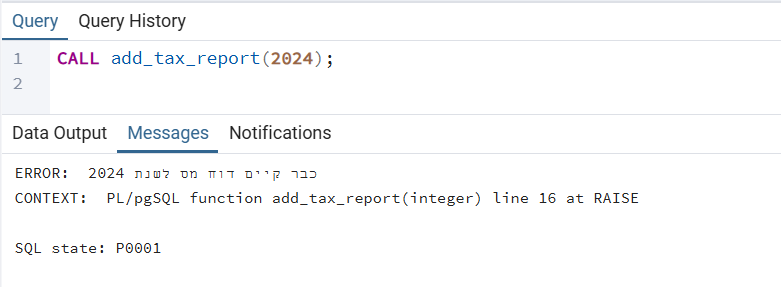
יצירת הפרוצדורה:



נתונים לפני הרצה:



ניסינו להריץ לקיבלנו:



ראינו כי קיימים כבר נתונים רבים אז עשינו מחיקה של דו"ח אחד וכן כל הפרטים המקושרים אליו על מנת לראות שזה עובד:

-- 1. auditrecord תלויה ב-expense וב-revenue

DELETE FROM auditrecord

WHERE expense\_id IN (

SELECT expense\_id FROM expense WHERE year = 2024

)

OR revenue\_id IN (

SELECT revenue\_id FROM revenue WHERE year = 2024

);

-- 2. invoice תלויה ב-payment

DELETE FROM invoice

WHERE payment\_id IN (

SELECT payment\_id FROM payment

WHERE revenue\_id IN (

SELECT revenue\_id FROM revenue WHERE year = 2024

)

);

-- 3. payment (תלויה ב-revenue)

DELETE FROM payment

WHERE revenue\_id IN (

SELECT revenue\_id FROM revenue WHERE year = 2024

);

-- 4. expense

DELETE FROM expense

WHERE year = 2024;

-- 5. revenue

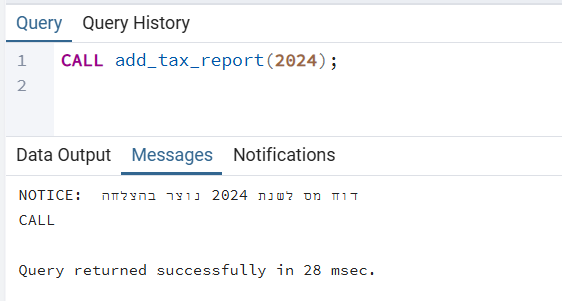
DELETE FROM revenue

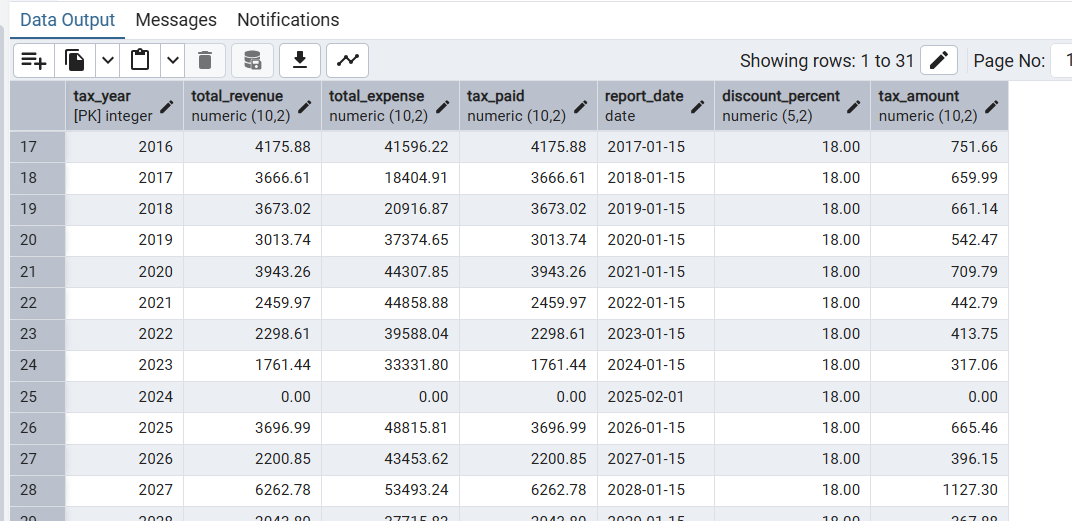
WHERE year = 2024;

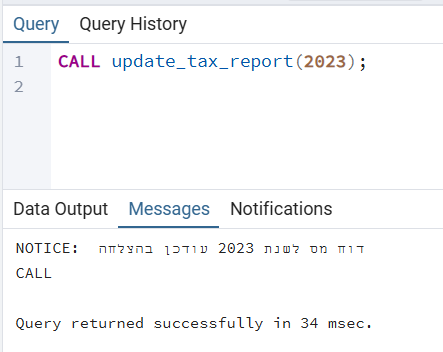
-- 6. לבסוף: taxreport

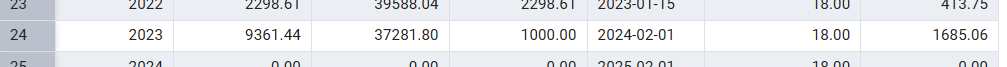
DELETE FROM taxreport

WHERE tax\_year = 2024;



אכן עבד אך זה מה שקיבלנו כי מחקנו את כל נתוני אותה השנה:  


שינינו את הפרוצדורה שתעדכן דו"ח קיים ןלא תיצור שורה חדשה:  
הוספנו נתונים על מנת לראות שינוי:

  
 והנה הדו"ח של 2023 מעודכן לאחר התוספות:

פרוצדורה 3:  
create\_full\_payment\_flow הפרוצדורה מדמה פעולה של "רישום עסקה" מלאה .

### קלטים:

* description (של ההכנסה)
* amount (של התשלום והחשבונית)
* payment\_method
* recipient (לקוח)
* issue\_date לחשבונית

התוצאה תהיה רישום ל־3 טבלאות עם קשרים מתאימים

CREATE OR REPLACE PROCEDURE create\_full\_payment\_flow(

p\_description VARCHAR,

p\_amount NUMERIC(10,2),

p\_payment\_method VARCHAR,

p\_recipient VARCHAR,

p\_issue\_date DATE

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_year INTEGER := EXTRACT(YEAR FROM p\_issue\_date)::INT;

v\_revenue\_id INTEGER;

v\_payment\_id INTEGER;

v\_invoice\_id INTEGER;

BEGIN

-- 1. יצירת הכנסה וקבלת ה-ID החדש

INSERT INTO revenue (description, year)

VALUES (p\_description, v\_year)

RETURNING revenue\_id INTO v\_revenue\_id;

-- 2. יצירת תשלום וקבלת ה-ID החדש

INSERT INTO payment (payment\_method, payment\_date, amount, revenue\_id)

VALUES (p\_payment\_method, p\_issue\_date, p\_amount, v\_revenue\_id)

RETURNING payment\_id INTO v\_payment\_id;

-- 3. יצירת חשבונית וקבלת ה-ID החדש

INSERT INTO invoice (recipient, issue\_date, payment\_id, amount)

VALUES (p\_recipient, p\_issue\_date, v\_payment\_id, p\_amount)

RETURNING invoice\_id INTO v\_invoice\_id;

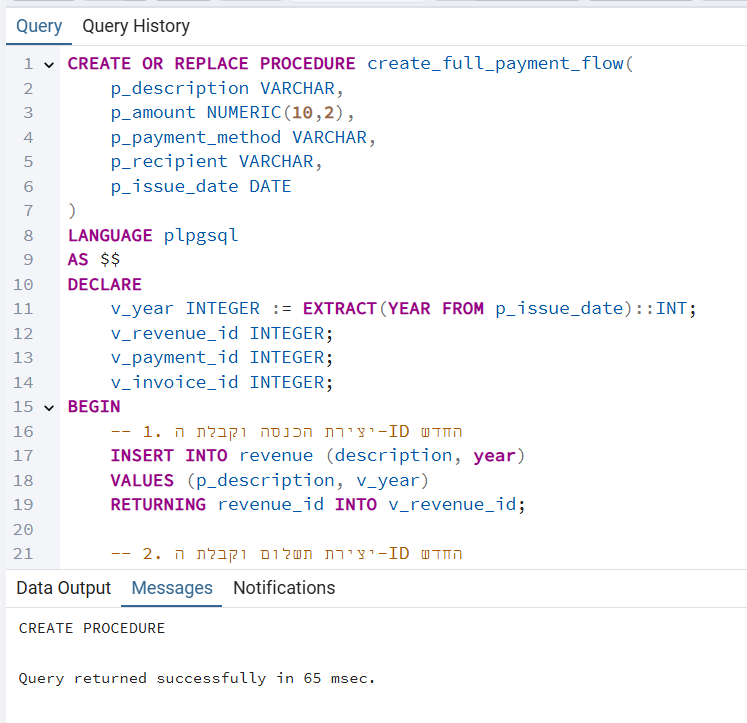
RAISE NOTICE 'תהליך הושלם: revenue\_id = %, payment\_id = %, invoice\_id = %',

v\_revenue\_id, v\_payment\_id, v\_invoice\_id;

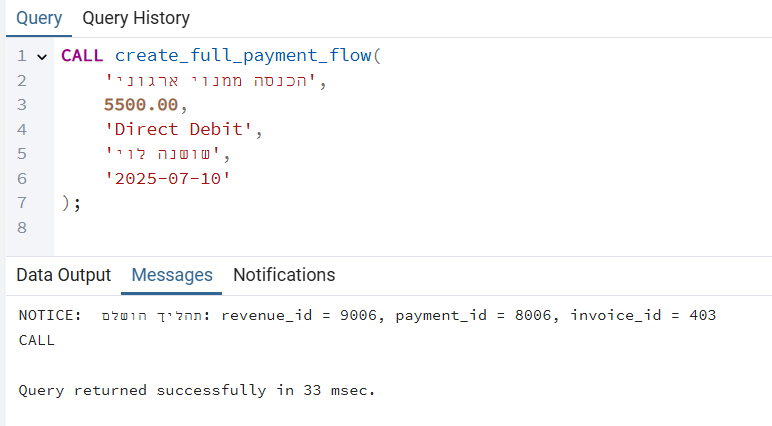
END;

$$;

יצירת הפרוצדורה:



הרצת הפרוצדורה:

****

**// תצלומי מסך של התוספות שהתווספו אוטומטית**

**טריגר 1:**

### מטרת הטריגר:

כשמוסיפים הוצאה (expense) מסוג **"Tax payment"**  
→ הוא יעדכן **במיידי** את שדה tax\_paid בטבלת taxreport  
→ כך שהשדה תמיד יציג את סכום המס ששולם בפועל עד כה באותה שנה

טריגר זה מתרחש בעת הכנסת הוצאה מסוג Tax payment ומעדכן את tax\_paid בדוח המס בהתאם

פונקציית הטריגר:  
CREATE OR REPLACE FUNCTION update\_tax\_paid\_on\_tax\_expense()

RETURNS TRIGGER AS

$$

DECLARE

v\_type\_name TEXT;

BEGIN

-- נוודא שסוג ההוצאה הוא באמת "Tax payment"

SELECT expense\_type INTO v\_type\_name

FROM expensetypes

WHERE type\_id = NEW.type\_id;

IF v\_type\_name = 'Tax payment' THEN

UPDATE taxreport

SET tax\_paid = COALESCE(tax\_paid, 0) + NEW.amount

WHERE tax\_year = NEW.year;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;  
  
  
יצירת הטריגר:  
CREATE TRIGGER trg\_update\_tax\_paid\_on\_tax\_expense

AFTER INSERT ON expense

FOR EACH ROW

EXECUTE FUNCTION update\_tax\_paid\_on\_tax\_expense();

בדיקה:

לפני:

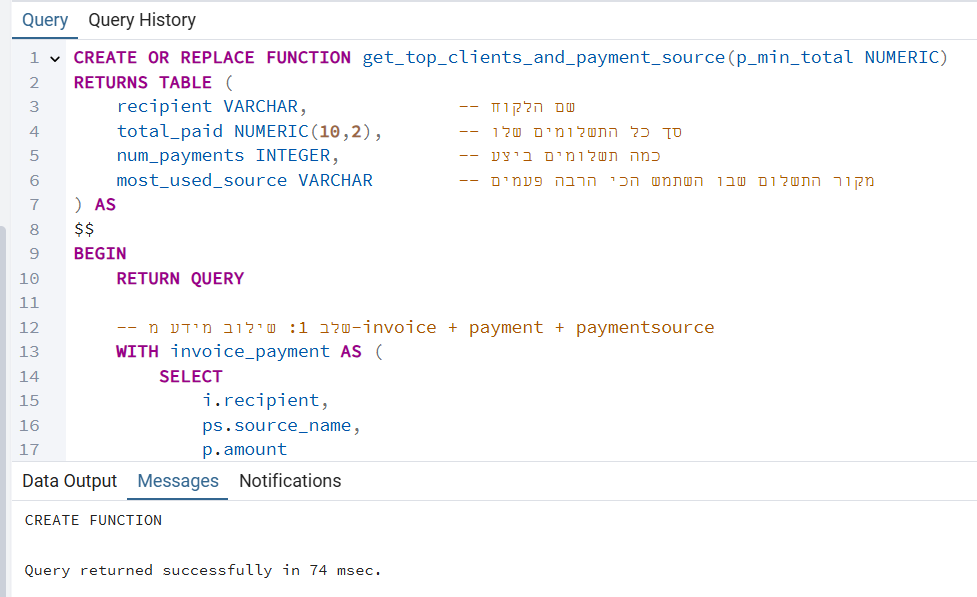


INSERT INTO expense (amount, status, due\_date, description, year, type\_id)

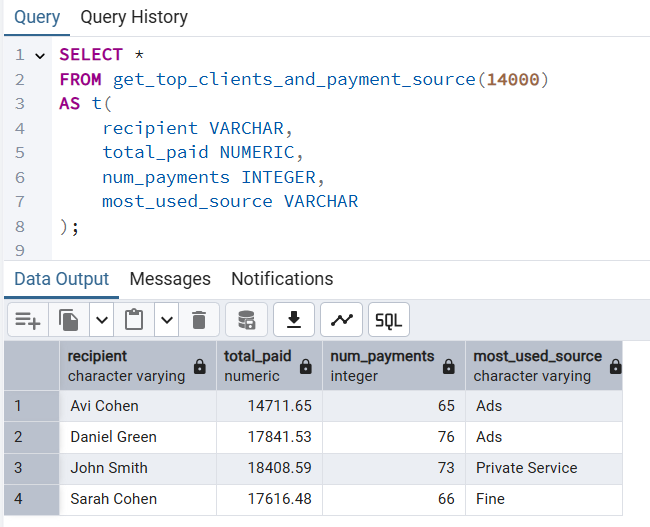
VALUES (405.00, 'Paid', '2023-12-30', 'תשלום מס', 2023, 16);

אחרי:  
  


**פונקציה 2:**

**יצירת הפונקציה:  
**

**דוגמא להפעלת הפונקציה:**

****

**טריגר 2:  
יצירת הטריגר:**

### תוכנית ראשית 1: תיאור התוכנית הראשית: טעינת נתונים והרצת דוחות

בפרויקט מימשנו תוכנית ראשית עשירה שתכליתה לדמות הפעלת מערכת אמיתית לאורך שנים רבות, תוך יצירת נתונים אוטומטית, עיבוד, חישוב והפקת דוחות תקופתיים. התוכנית כוללת ארבעה שלבים עיקריים:

#### שלב 1: הכנסת תשלומים והפקת חשבוניות

בשלב זה מבוצע שימוש בפרוצדורה create\_full\_payment\_flow, אשר מדמה תשלום בפועל על ידי לקוח, תוך יצירת רשומת הכנסה (Revenue), תשלום (Payment), וחשבונית (Invoice) עבור כל פעולה. הנתונים נבחרים באופן אקראי מתוך מערכים שהוגדרו מראש עבור שמות לקוחות, אמצעי תשלום וסוגי הכנסות. כך מתקבל מגוון רחב של תשלומים אמיתיים.

#### שלב 2: הכנסת הוצאות שונות

המערכת מוסיפה באופן אוטומטי הוצאות שונות על פני השנים 2004–2024. סוג ההוצאה נבחר מתוך טבלת expensetypes (כגון דלק, תוכנה, שכר עובדים וכדומה), והנתונים נשמרים בטבלת expense. אם סכום ההוצאה גבוה מ־10,000 ₪ מופעל טריגר ליצירת תיעוד בטבלת ביקורת.

#### שלב 3: עדכון דוחות מס שנתיים

התוכנית קוראת לפרוצדורה update\_tax\_report עבור כל שנה בין 2004 ל־2024. פרוצדורה זו מחשבת את סך ההכנסות וההוצאות באותה שנה, ומעדכנת את שדה tax\_paid בהתאם לסכומי ההוצאה מסוג "Tax Payment".

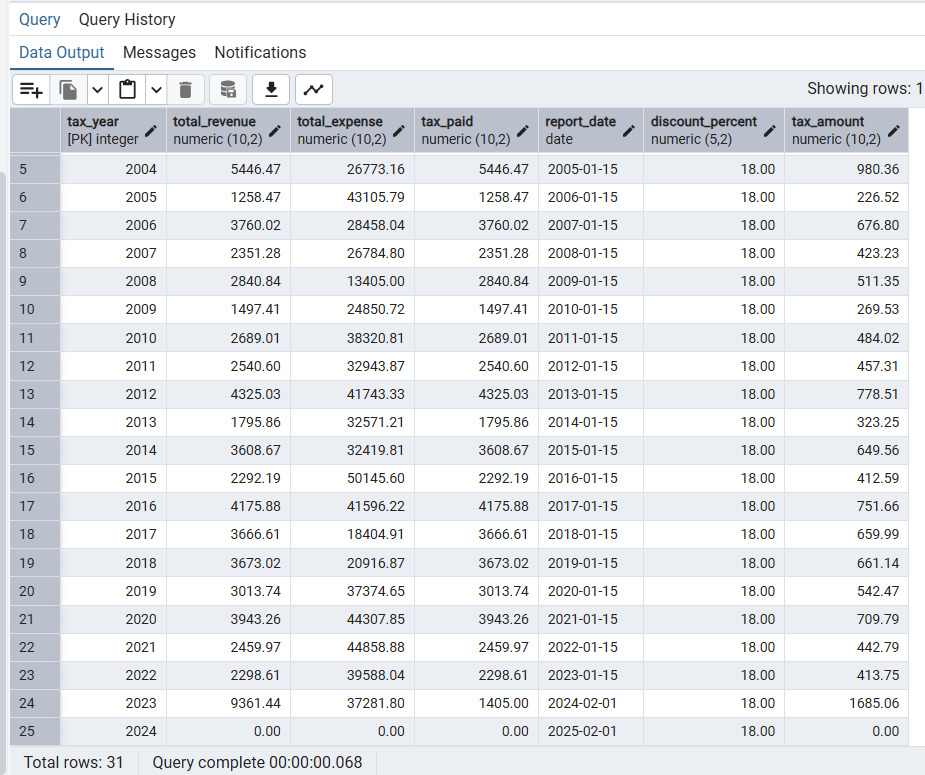
#### שלב 4: הצגת סיכומי הכנסות ותשלומים

לסיום, התוכנית מציגה את סיכום ההכנסות ומספר התשלומים שבוצעו עבור כל אחת מהשנים, באמצעות קריאה לפונקציה get\_yearly\_revenue\_summary. הפלט מוצג באמצעות הודעות NOTICE במסוף.

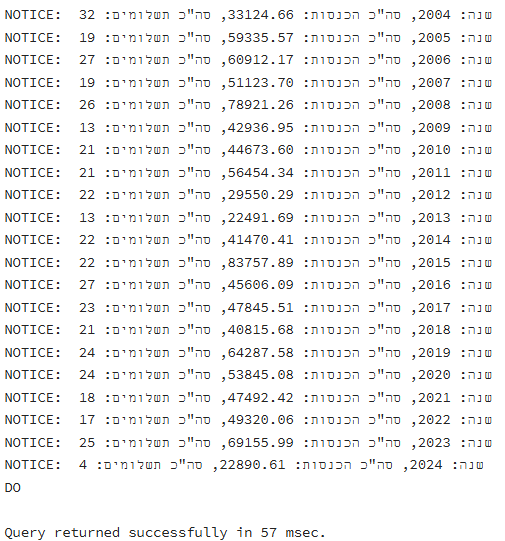
### מטרות ותרומה לפרויקט

תוכנית זו מאפשרת:

* בדיקה של תקינות כלל המודולים במערכת.
* יצירת נפח נתונים ריאליסטי לצורכי דמו, בדיקות והדמיות.
* הדגמה של אינטגרציה בין פונקציות, פרוצדורות, טריגרים וטבלאות.

**לפני הרצה:** **לאחר הרצה:**

****

**וכן חלק מהפלט:  
**

### תוכנית ראשית 2: תיאור התוכנית הראשית השנייה:

**מטרת התוכנית:**  
לזהות לקוחות מרכזיים שביצעו תשלומים משמעותיים לחברה (מעל לסכום סף נתון), ולהציג נתונים סטטיסטיים על פעילותם לאחר עדכון סטטוס החשבוניות.

**🔹 שלב 1: הפעלת הפרוצדורה update\_paid\_invoices()**

**מה היא עושה?**  
הפרוצדורה מבצעת סריקה על כל החשבוניות בטבלת Invoice ומשווה בין הסכום שהוגדר בחשבונית לבין הסכום ששולם בפועל (Payment). אם הסכומים זהים – היא מעדכנת את סטטוס החשבונית ל־Paid.  
זוהי פרוצדורה חשובה לשמירה על עקביות נתונים בדוחות.

**🔹 שלב 2: שימוש בפונקציה get\_top\_clients\_and\_payment\_source(p\_min\_total NUMERIC)**

**מה היא עושה?**  
הפונקציה מחזירה רשימה של לקוחות ששילמו סכום כולל גבוה מסף מסוים (לדוגמה: ₪10,000).  
הפלט כולל:

* שם הלקוח (recipient)
* סך התשלומים שביצע (total\_paid)
* מספר התשלומים (num\_payments)
* מקור התשלום בו השתמש הכי הרבה (most\_used\_source)

**טכניקות עיקריות:**

* שימוש ב־ROW\_NUMBER() כדי לזהות את מקור התשלום העיקרי.
* חלוקה לתת-שלבים פנימיים באמצעות WITH (Common Table Expressions) לשמירה על קריאות הקוד.

**🔹 שלב 3: הדפסת הפלט**

לאחר הפעלת הפונקציה, מתקבלות רשומות שהמערכת מדפיסה באמצעות RAISE NOTICE.  
כך ניתן לעקוב אחר פעילות הלקוחות המרכזיים במערכת ולהסיק מסקנות עסקיות או שיווקיות.

