ב"ה

**מיני פרויקט בבסיסי נתונים:**

ניהול תרומות ליולדות

מגישות:

אילה גליס - 324128354

חלי עדני הכהן - 206441479

תוכן ענינים:

[**תיאור הפרויקט**](#_j6m239bg7nph) 3

**ERD - Diagram.............................................................................................**4

[**DSD - Diagram**](#_miu9iq8l8mhm) 5

[**פקודות ה create table**](#_9h0r07puztfp) 6

[**פקודת ה Desc**](#_fs7zkzgvaa8) 9

[**הכנסת נתונים על ידי Data Generator**](#_9o8m6zbwu7f9) 10

[**הכנסת נתונים על ידי קובץ text**](#_4rbhqlomi5tp) 16

[**הכנסת נתונים עלי ידי סקריפט פיתון**](#_r2jewsjy1723) 19

[**גיבוי ושחזור**](#_ny7czw50241t) 21  
שינויים בפרויקט..............................................................................................................................22

Constraints..............................................................................................................................26

שאילתות Select............................................................................................................................27

שאילתות Update..........................................................................................................................31

שאילתות Delete............................................................................................................................33

שאילתות עם פרמטרים.....................................................................................................................35

פונקציה 1.....................................................................................................................................37

פרוצדורה 1...................................................................................................................................39

תוכנית 1.......................................................................................................................................41

פונקציה ........................................................................................................................................42

פרוצדורה 2 ..................................................................................................................................43

תוכנית 2 .......................................................................................................................................45

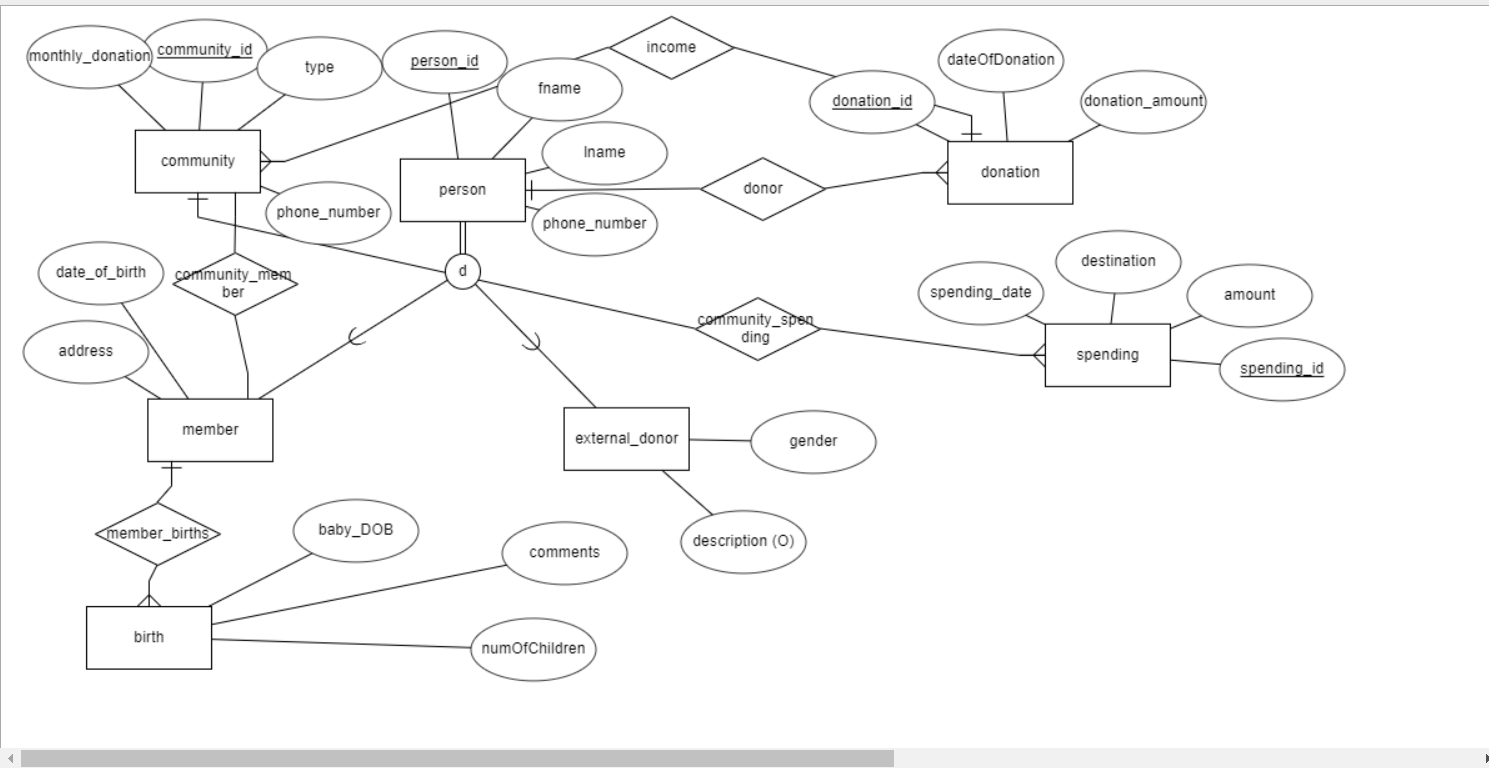
**מערכת לניהול תרומות:**

המערכת שלנו עוסקת בניהול תרומות לקהילות שמחלקות את הכסף ליולדות.

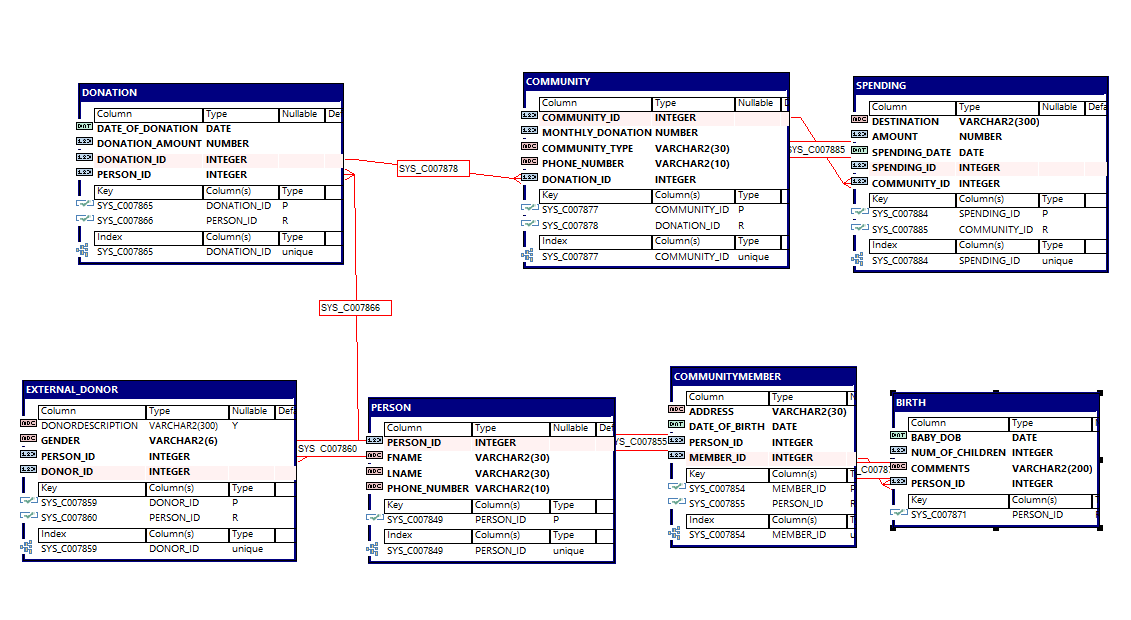
המערכת כוללת את הישויות הבאות: אדם כללי, חבר בקהילה, תורם חיצוני, תרומה לקהילה, לידה, קהילה, והוצאות של הקהילה.

יש כל מיני סוגים של קהילות ובכל קהילה יש חברים שתורמים לקהילה. בנוסף, יתכנו גם תורמים חיצוניים. כל הכספים הנתרמים לקהילה מועברים ליולדות הזקוקות לעזרה.

**ERD Diagram**



**DSD-Diagram**

****

**Create Table**

**CREATE TABLE person**

**(**

**person\_id INT NOT NULL,**

**fname VARCHAR2(30) NOT NULL,**

**lname VARCHAR2(30) NOT NULL,**

**phone\_number VARCHAR2(10) NOT NULL,**

**PRIMARY KEY (person\_id)**

**);**

**CREATE TABLE communityMember**

**(**

**address VARCHAR2(30) NOT NULL,**

**date\_of\_birth DATE NOT NULL,**

**person\_id INT NOT NULL,**

**member\_id INT NOT NULL,**

**PRIMARY KEY (member\_id),**

**FOREIGN KEY (person\_id) REFERENCES person(person\_id)**

**);**

**CREATE TABLE external\_donor**

**(**

**donorDescription VARCHAR2(300),**

**gender VARCHAR2(6) NOT NULL,**

**person\_id INT NOT NULL,**

**donor\_id INT NOT NULL,**

**PRIMARY KEY (donor\_id),**

**FOREIGN KEY (person\_id) REFERENCES person(person\_id)**

**);**

**CREATE TABLE donation**

**(**

**date\_of\_donation DATE NOT NULL,**

**donation\_amount NUMBER NOT NULL,**

**donation\_id INT NOT NULL,**

**person\_id INT NOT NULL,**

**PRIMARY KEY (donation\_id),**

**FOREIGN KEY (person\_id) REFERENCES person(person\_id)**

**);**

**CREATE TABLE birth**

**(**

**baby\_DOB DATE NOT NULL,**

**num\_of\_children INT NOT NULL,**

**comments VARCHAR2(200) NOT NULL,**

**person\_id INT NOT NULL,**

**FOREIGN KEY (person\_id) REFERENCES communityMember(member\_id)**

**);**

**CREATE TABLE community**

**(**

**community\_id INT NOT NULL,**

**monthly\_donation NUMBER NOT NULL,**

**community\_type VARCHAR2(30) NOT NULL,**

**phone\_number VARCHAR2(10) NOT NULL,**

**donation\_id INT NOT NULL,**

**PRIMARY KEY (community\_id),**

**FOREIGN KEY (donation\_id) REFERENCES donation(donation\_id)**

**);**

**CREATE TABLE spending**

**(**

**destination VARCHAR2(300) NOT NULL,**

**amount NUMBER NOT NULL,**

**spending\_date DATE NOT NULL,**

**spending\_id INT NOT NULL,**

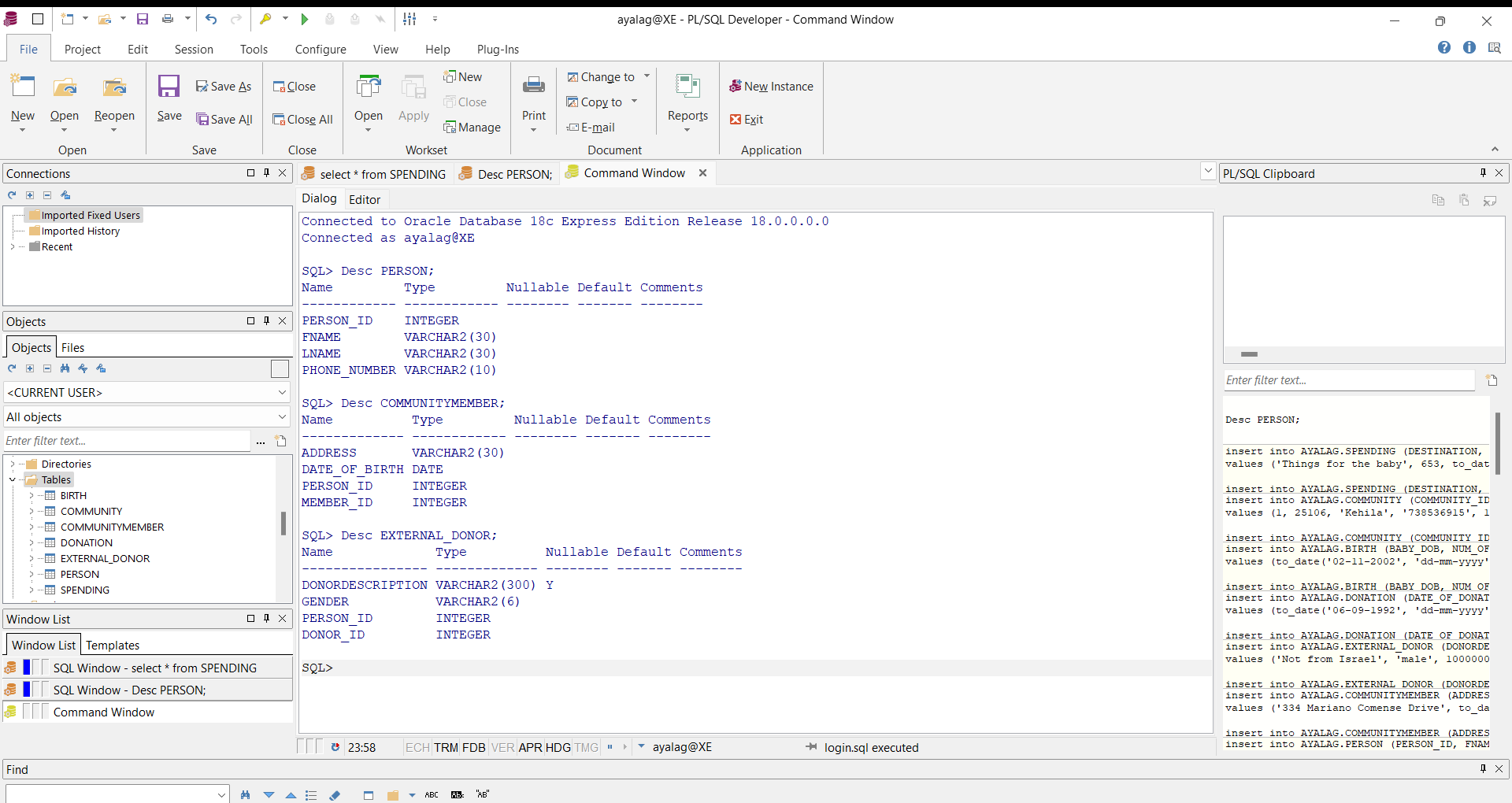
**community\_id INT NOT NULL,**

**PRIMARY KEY (spending\_id),**

**FOREIGN KEY (community\_id) REFERENCES community(community\_id)**

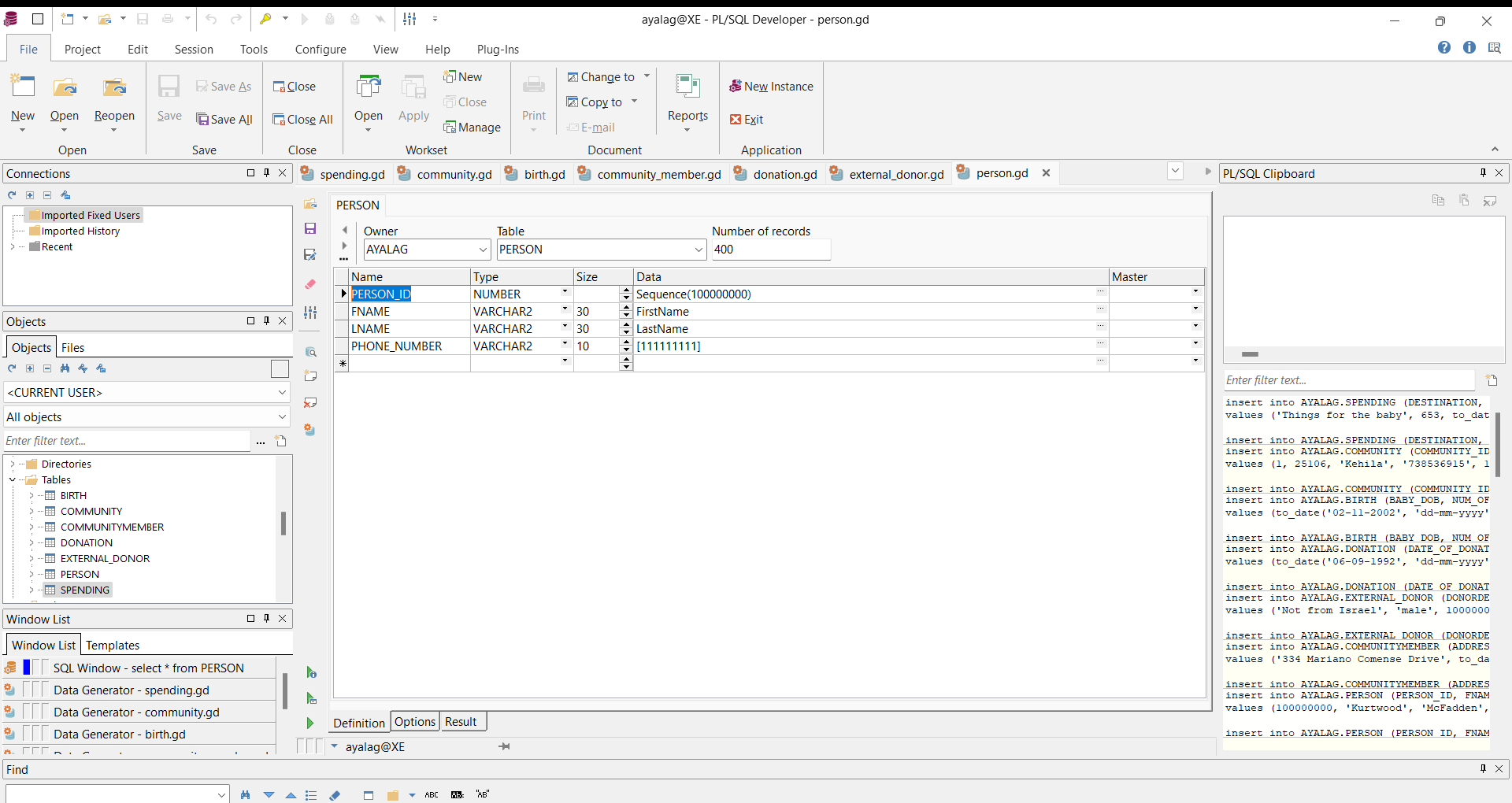
**);**

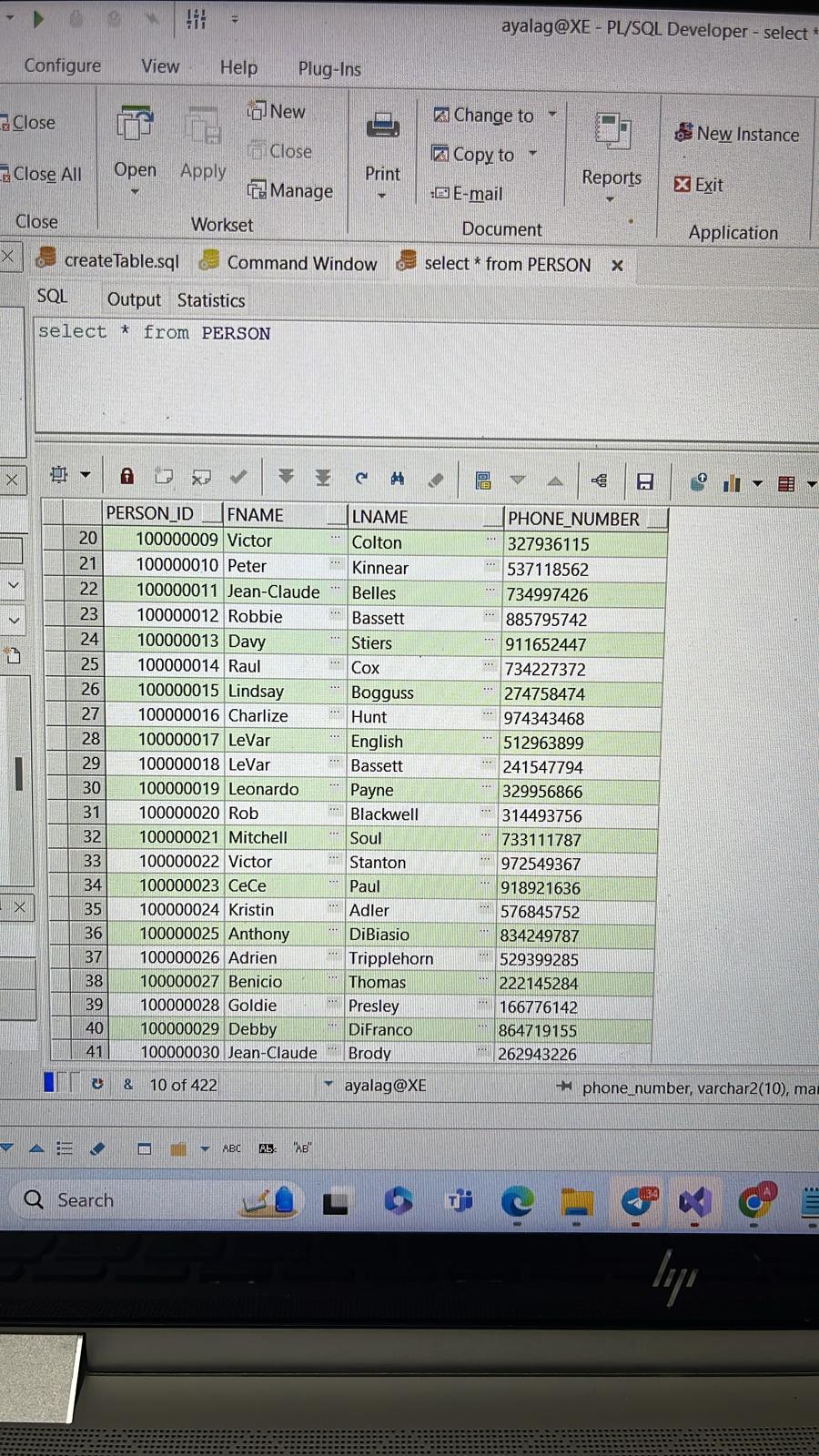
**Desc Command:**

****

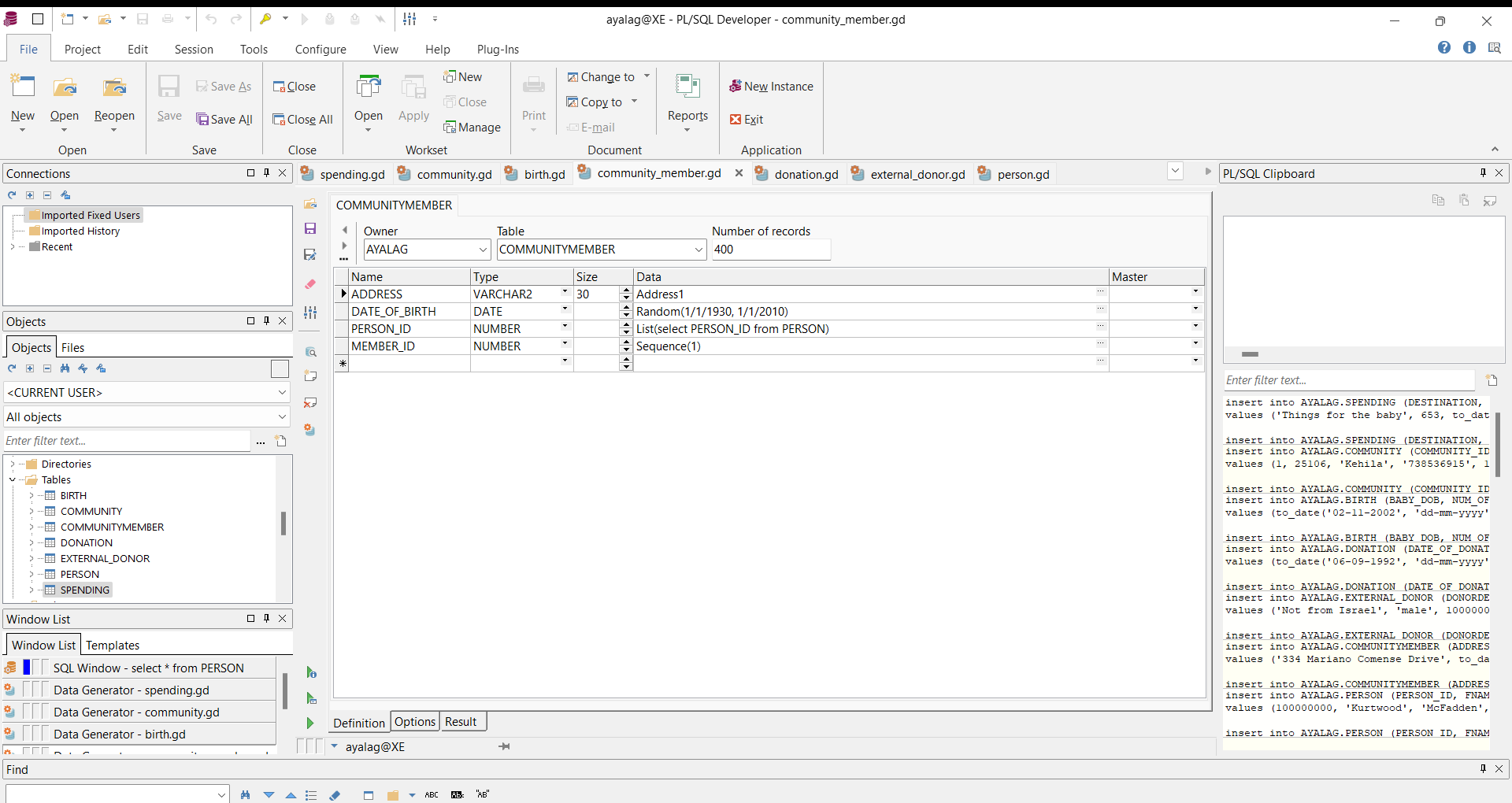
**Inserting data through data generator:**

**Person:**

****

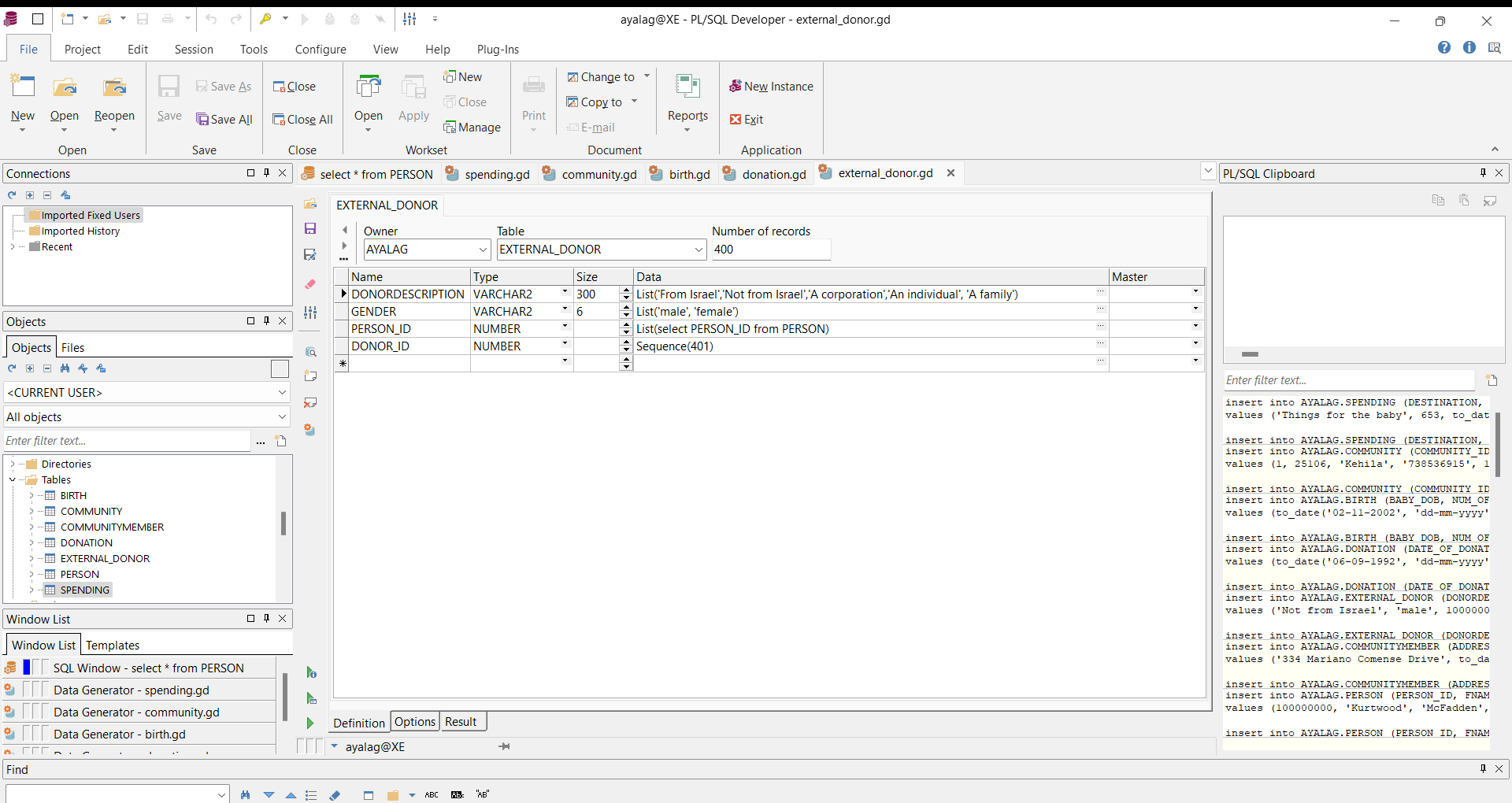
****

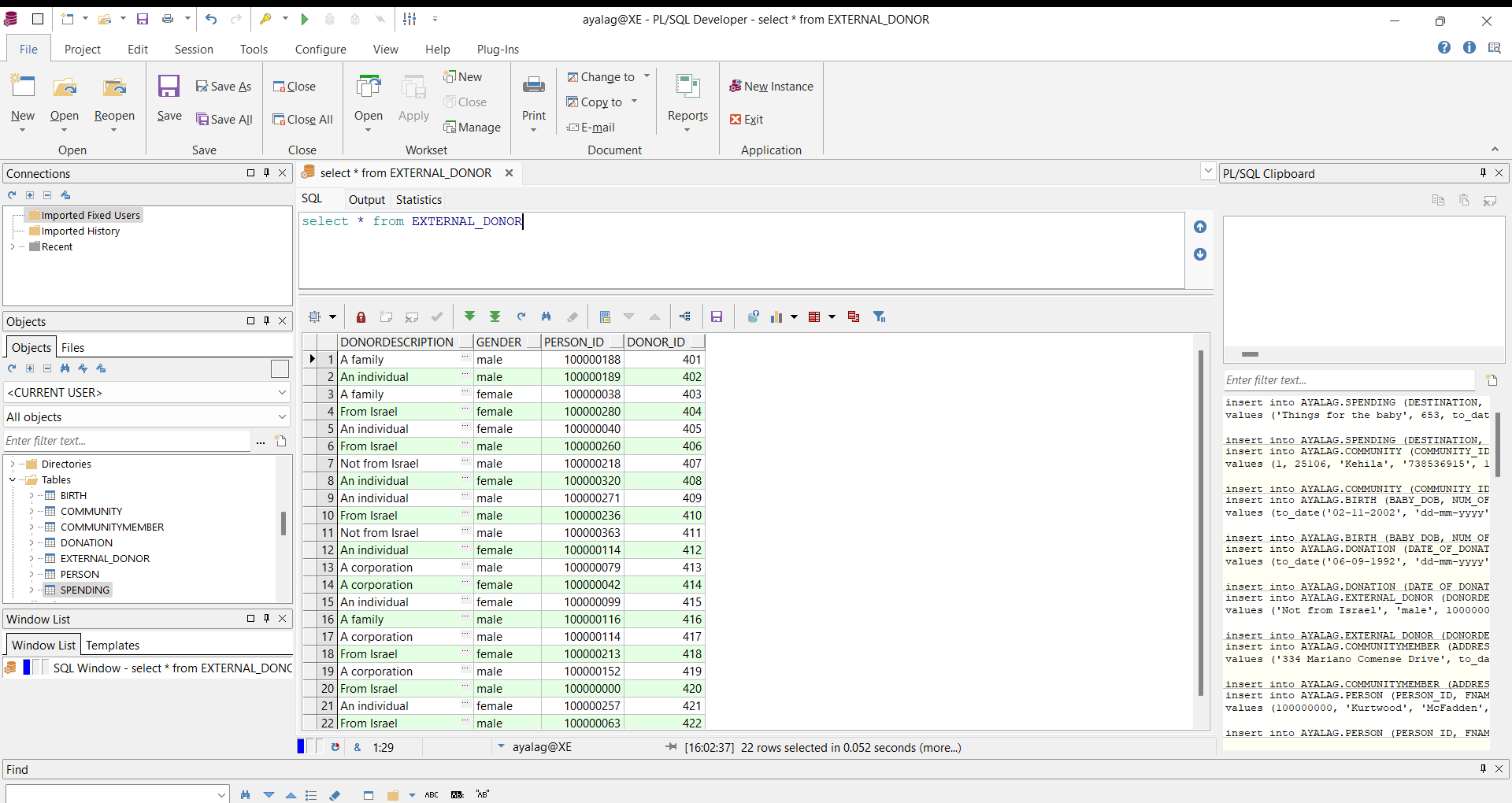
**Community Member:**

****

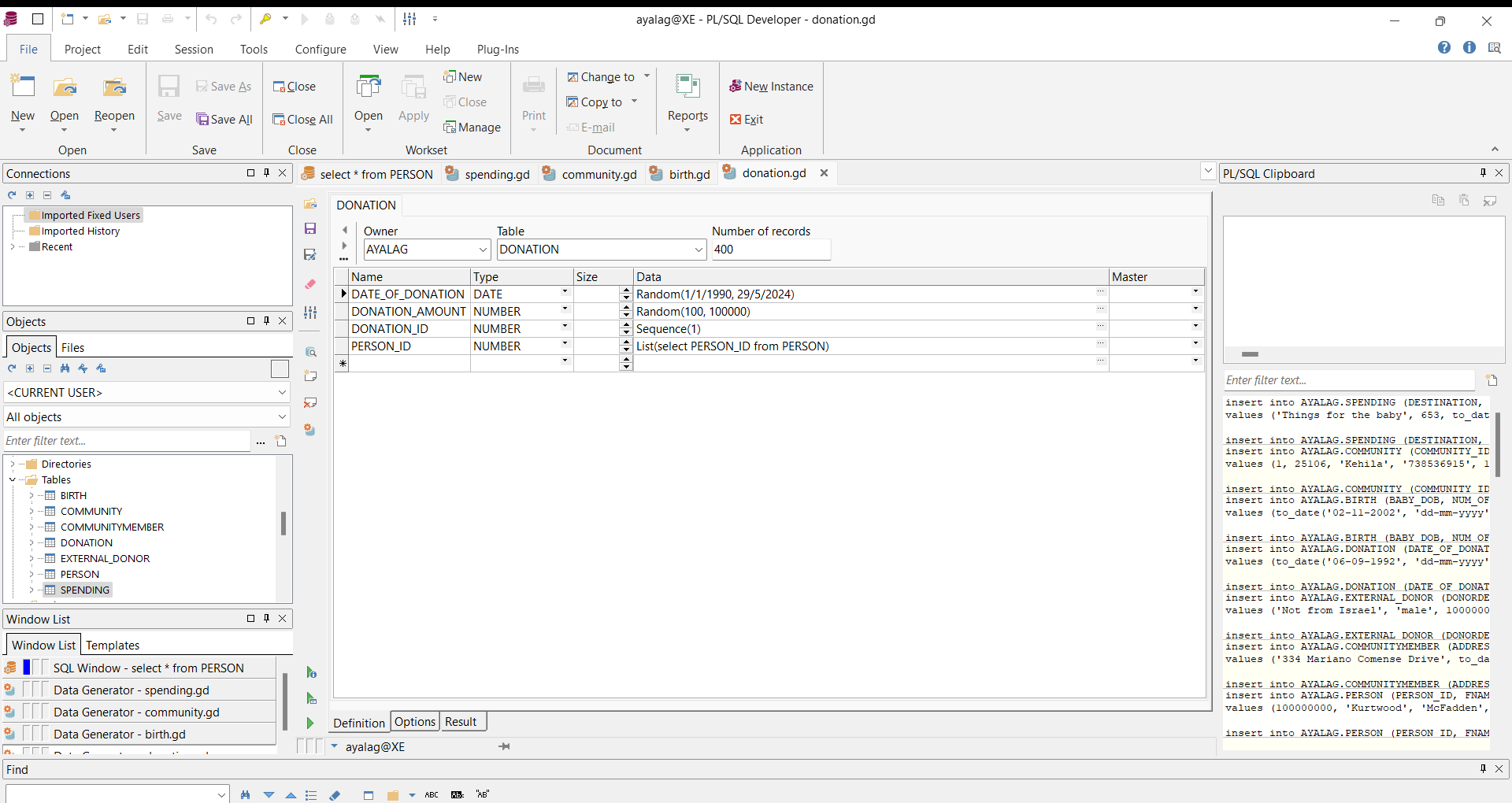
****

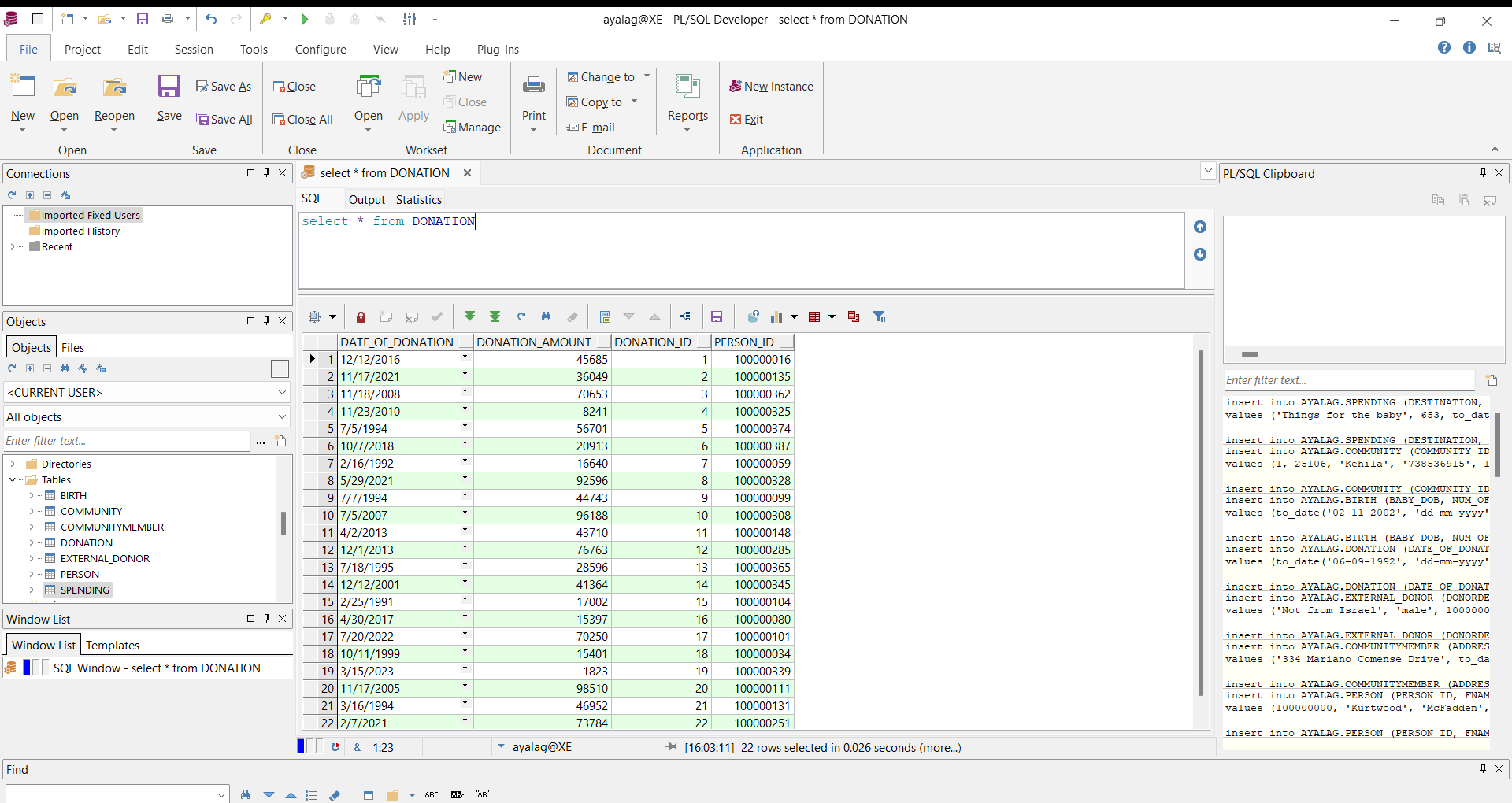
**External Donor:**

****

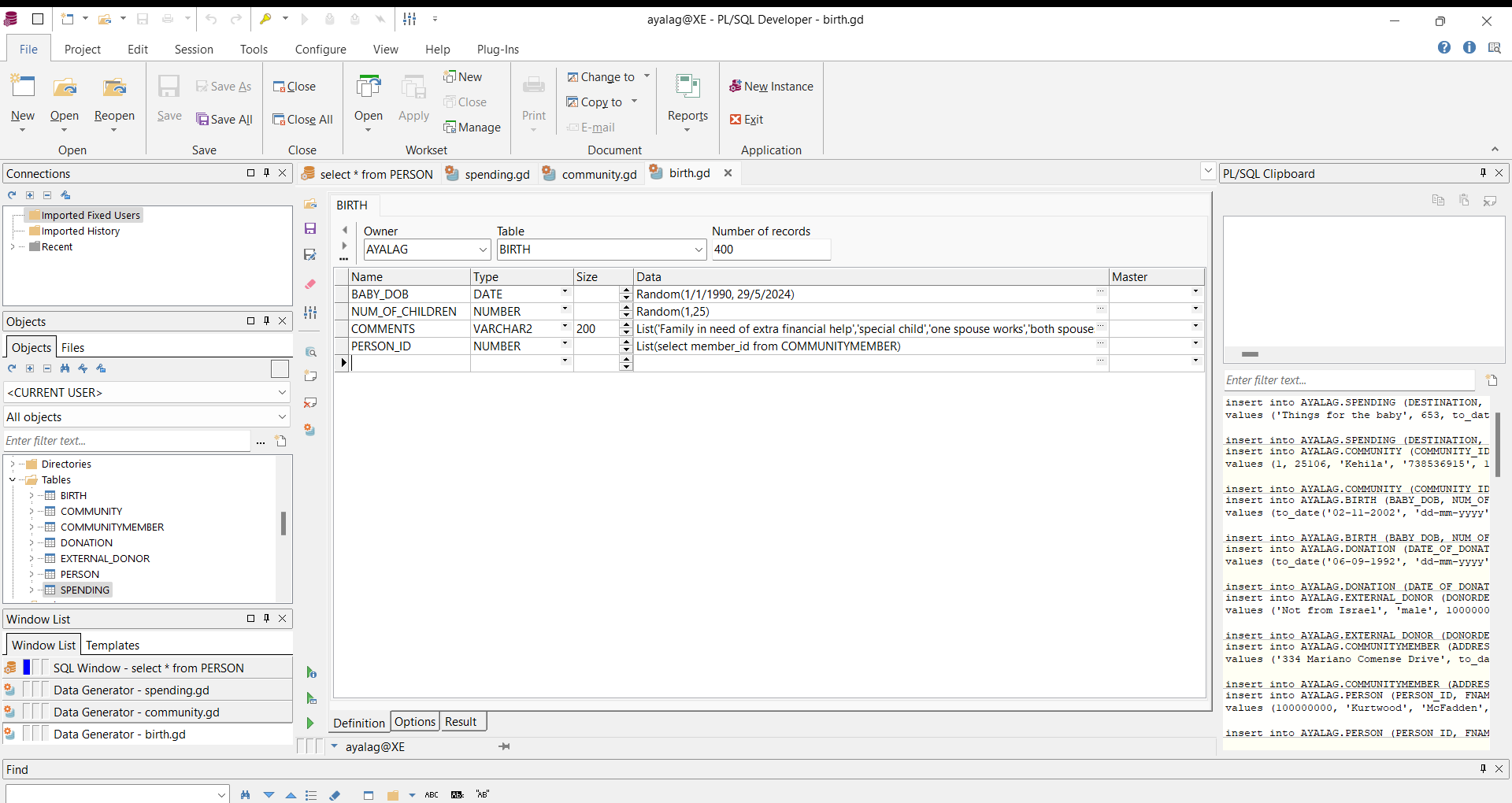
****

**Donation:**

****

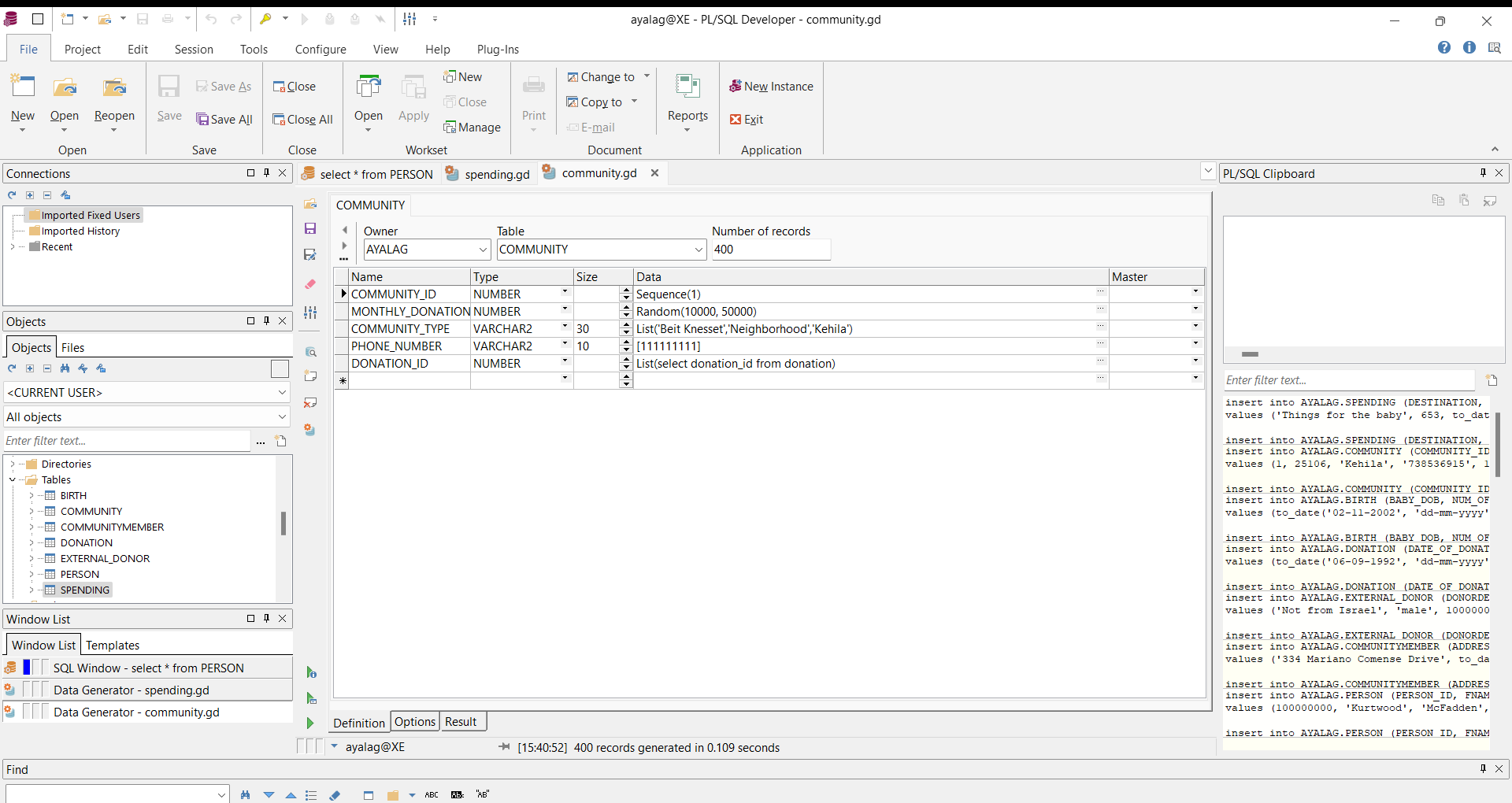
****

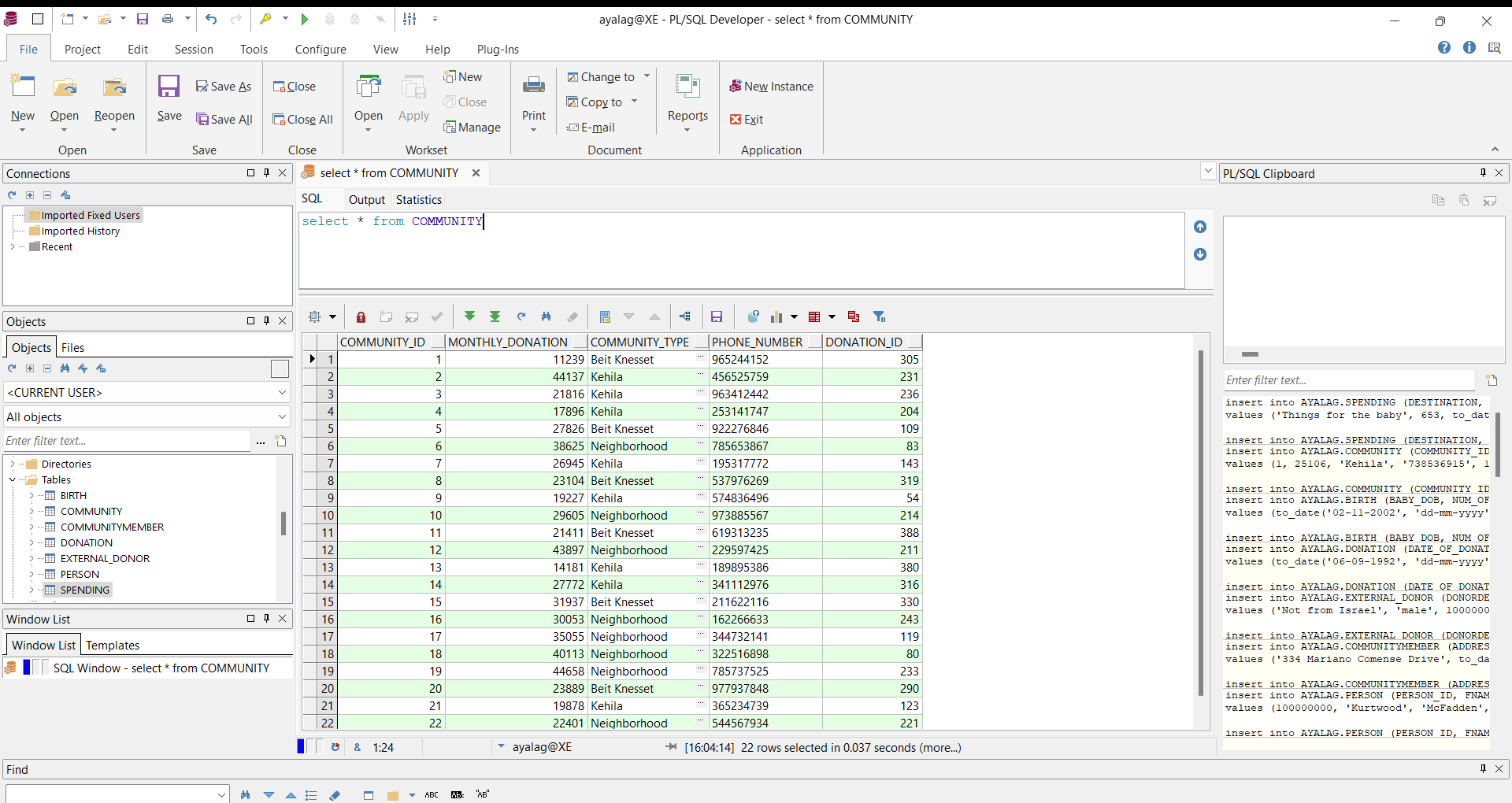
**Birth:**

****

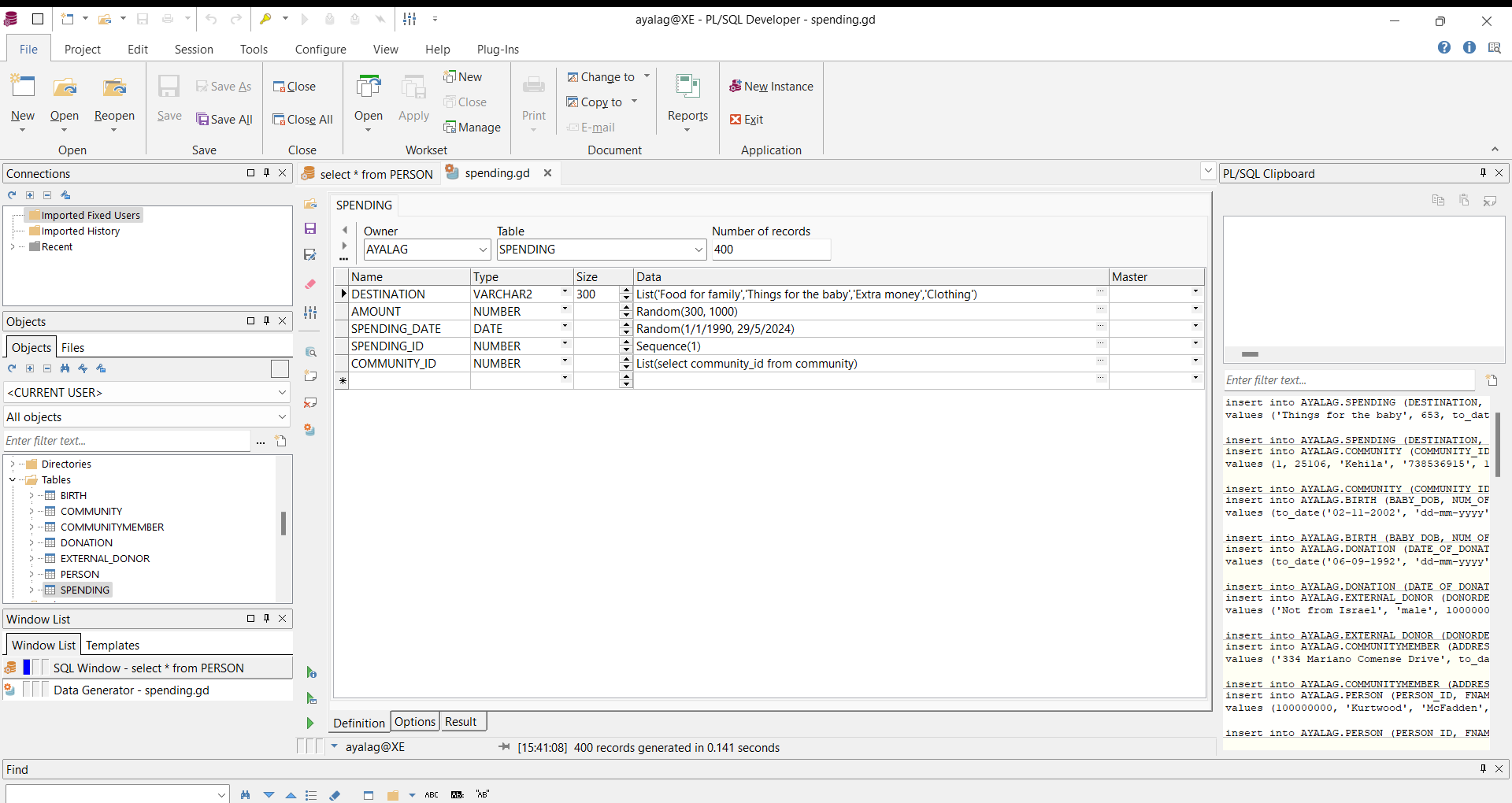
****

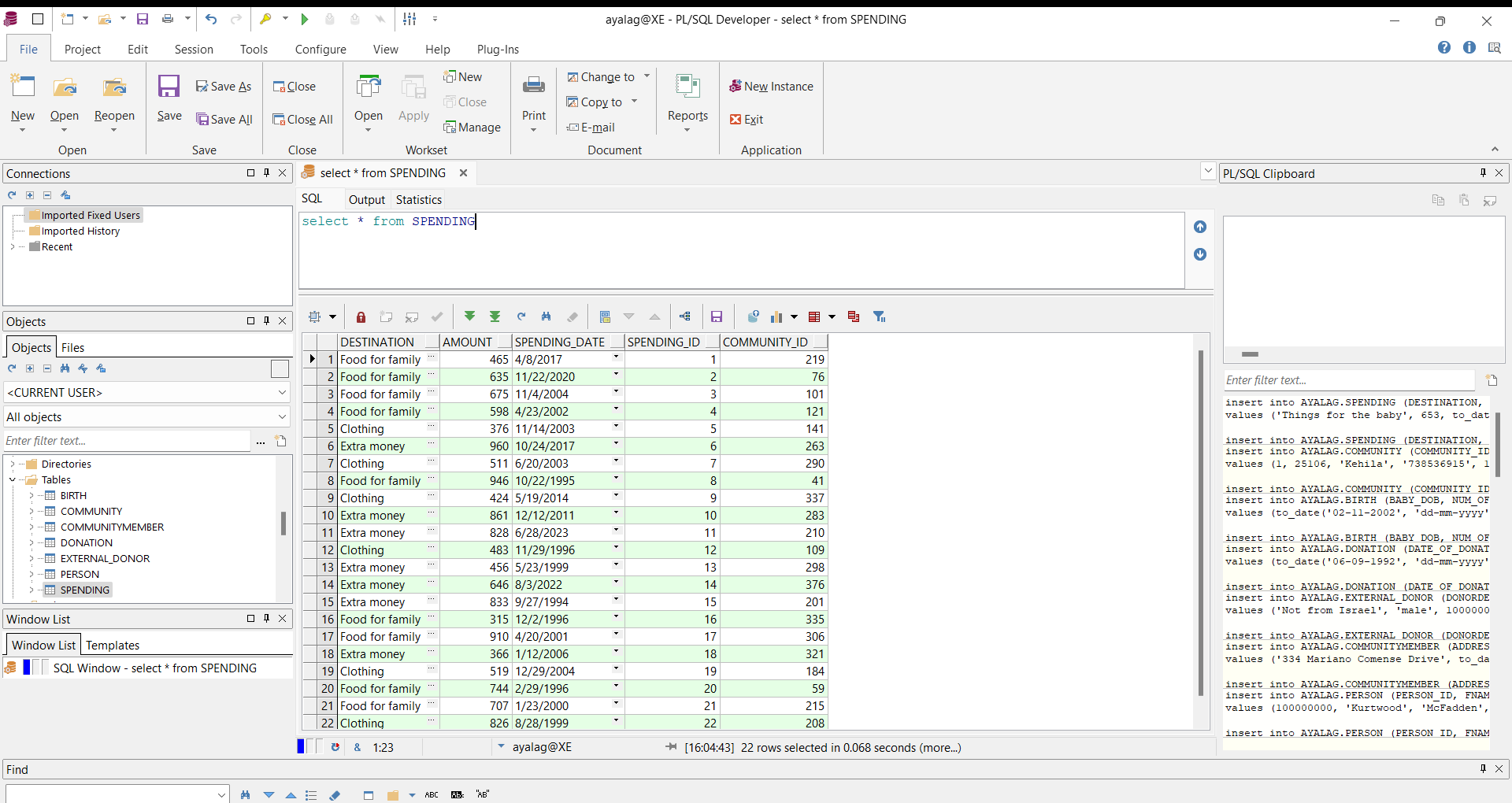
**Community:**

****

****

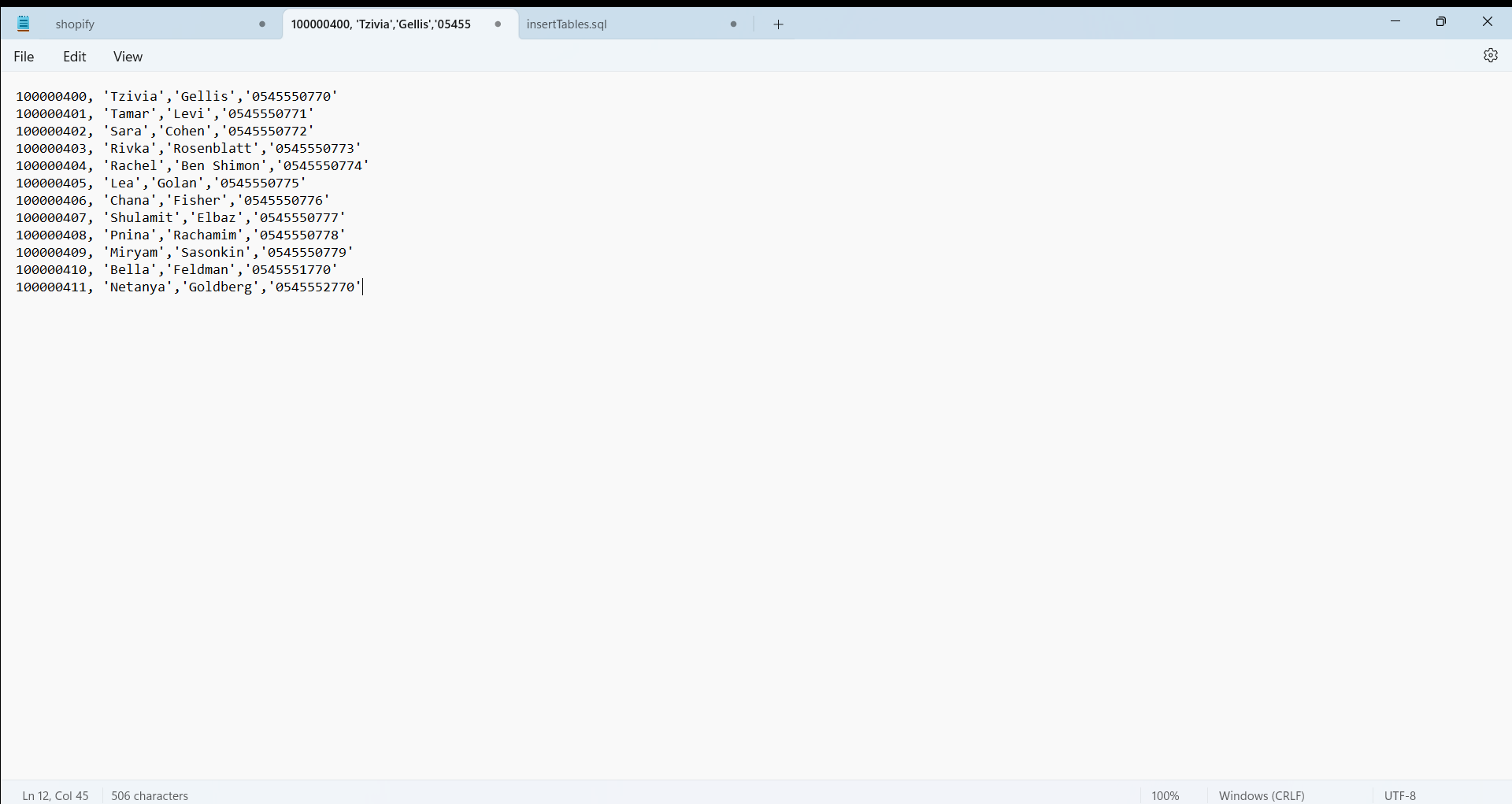
**Spending:**

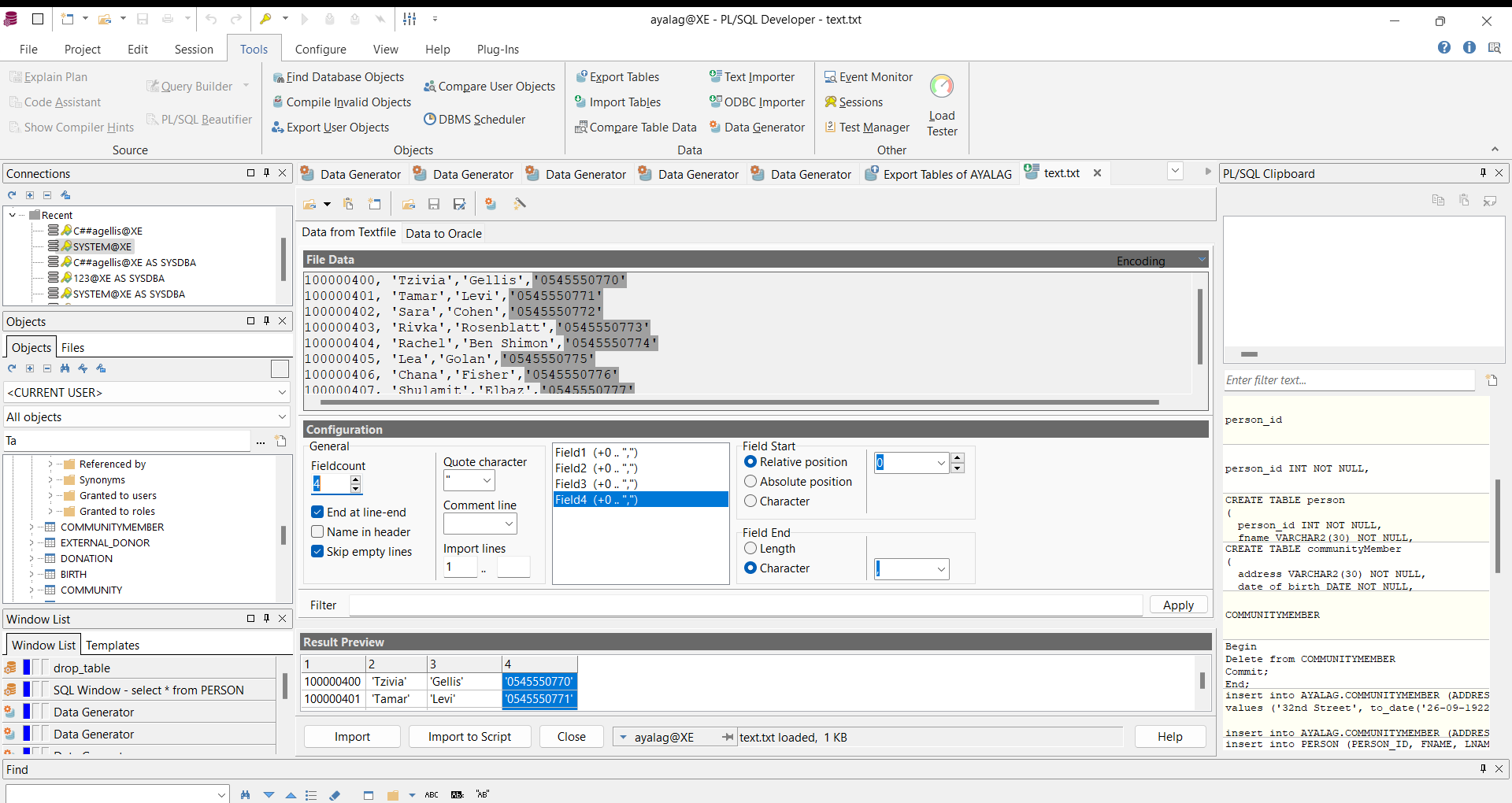
****

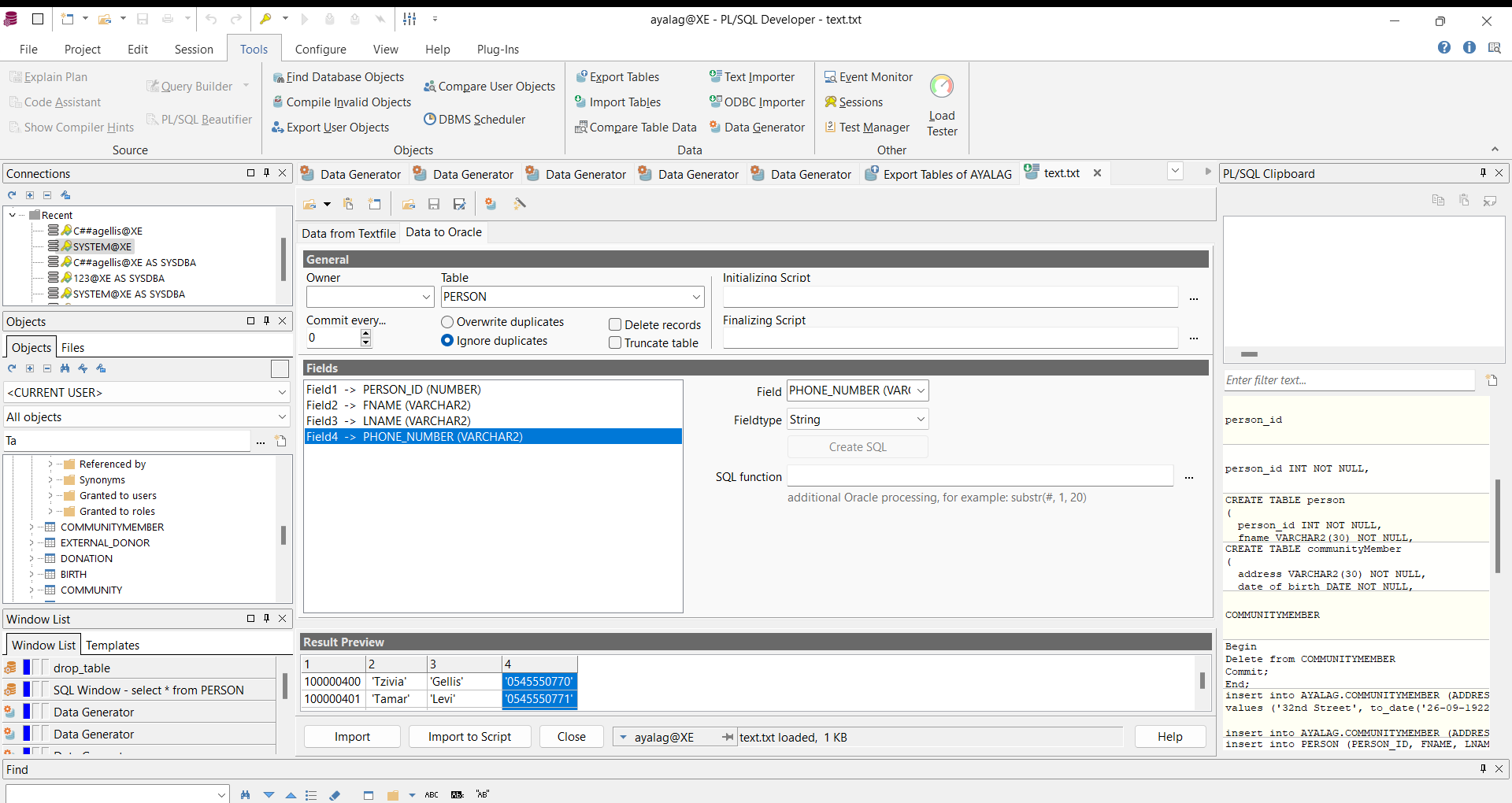
****

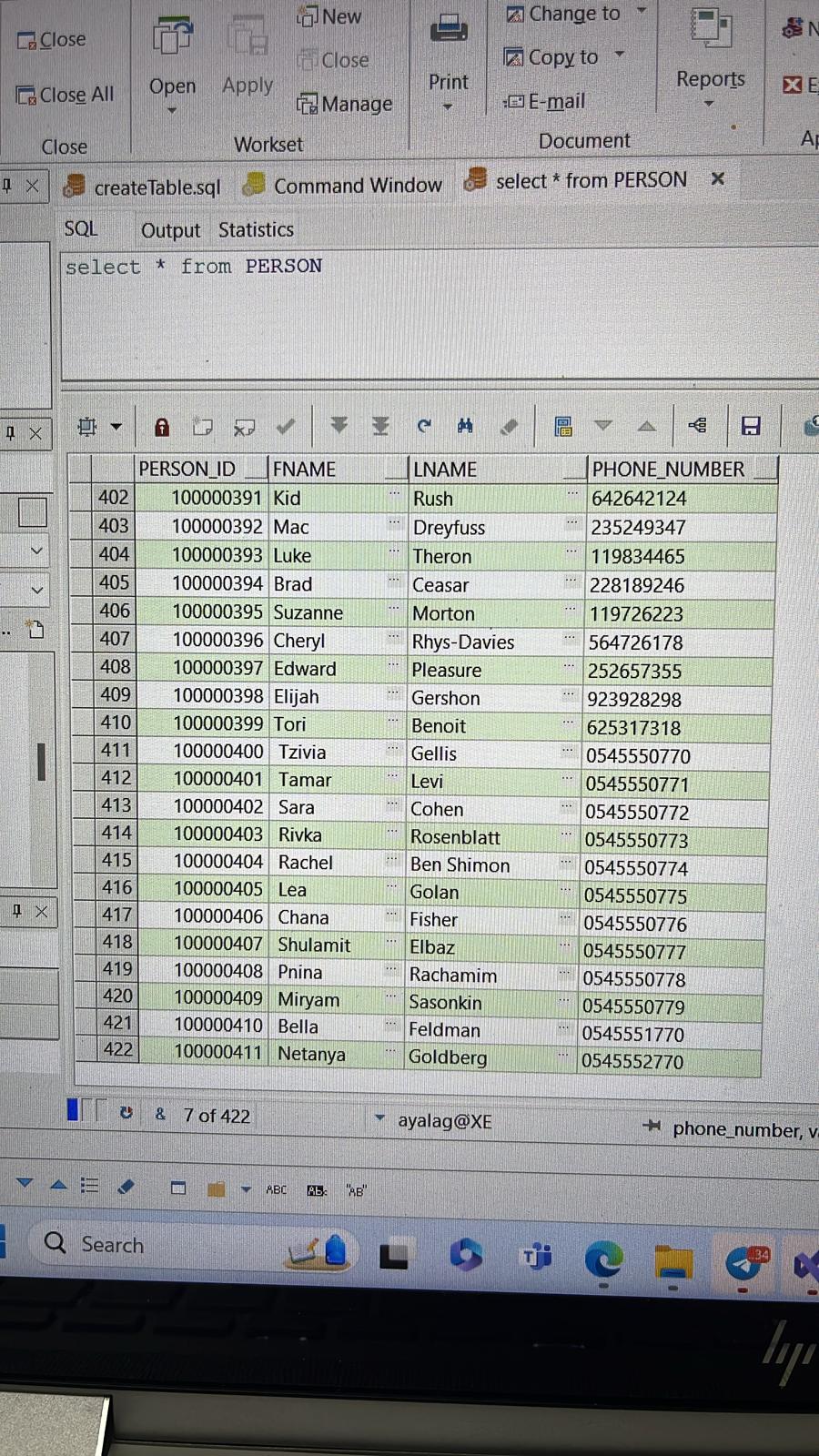
**Inserting data from a text file:**

**Person:**

****

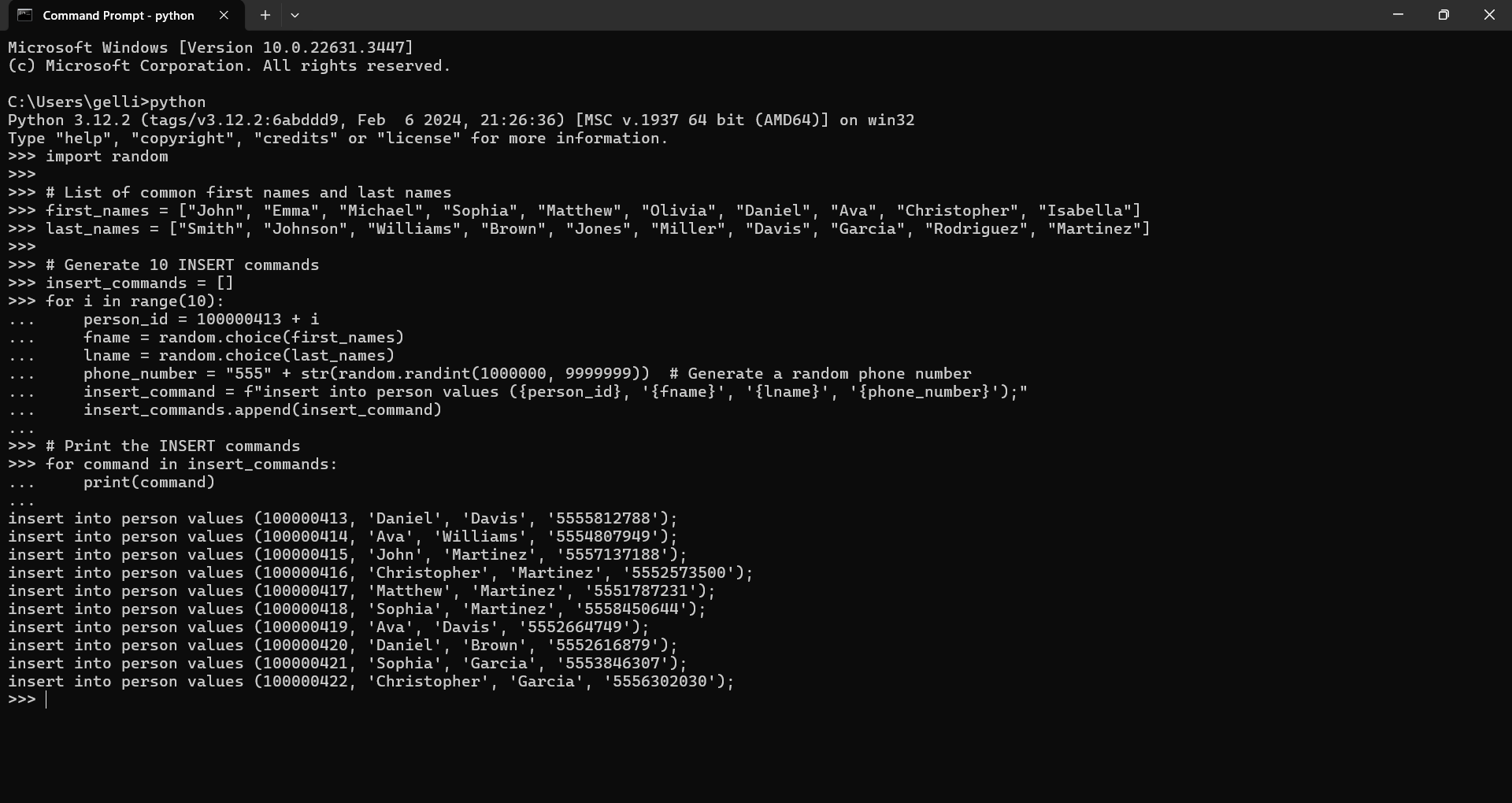
****

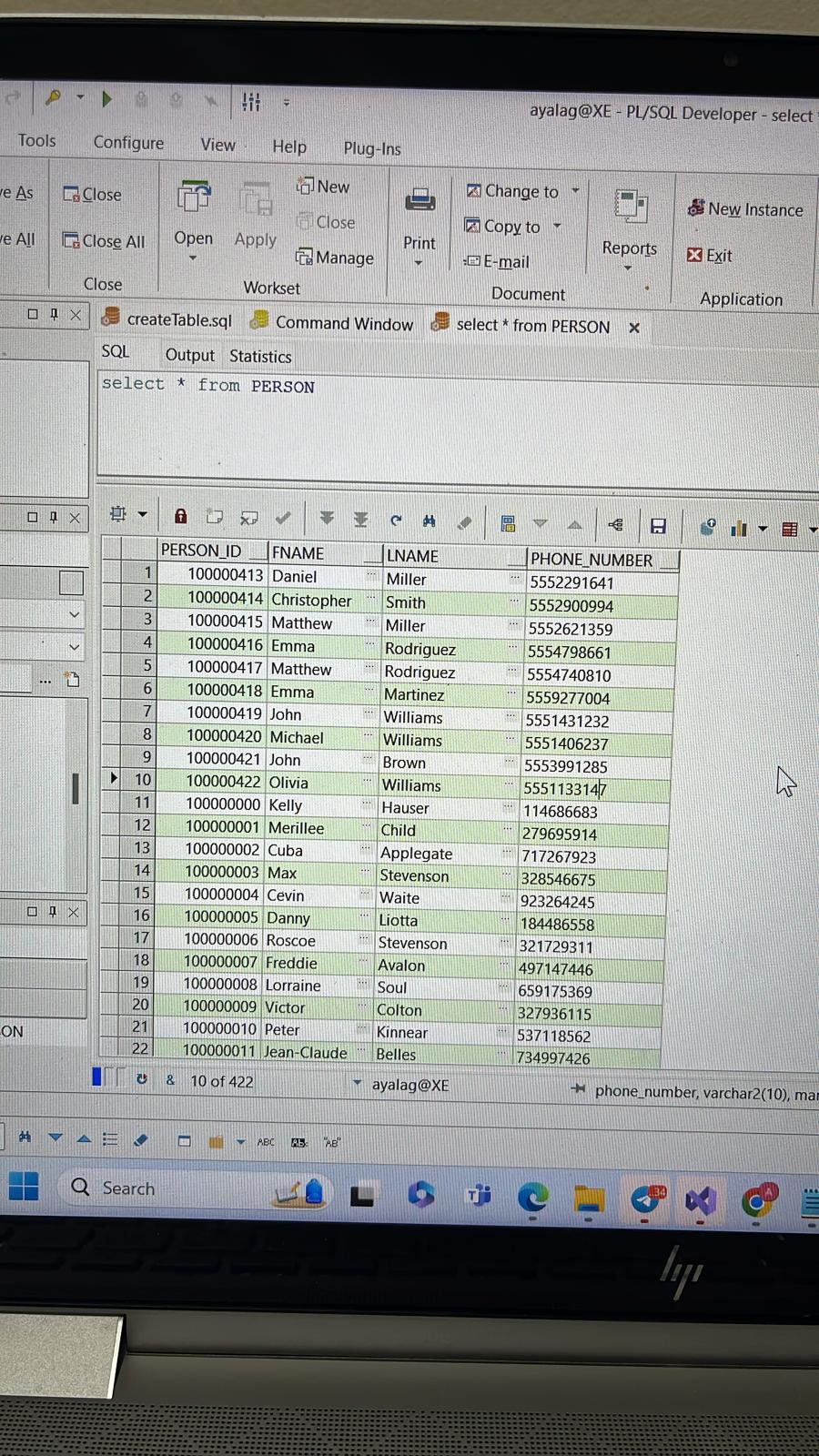
****

****

**Insert data by python script:**

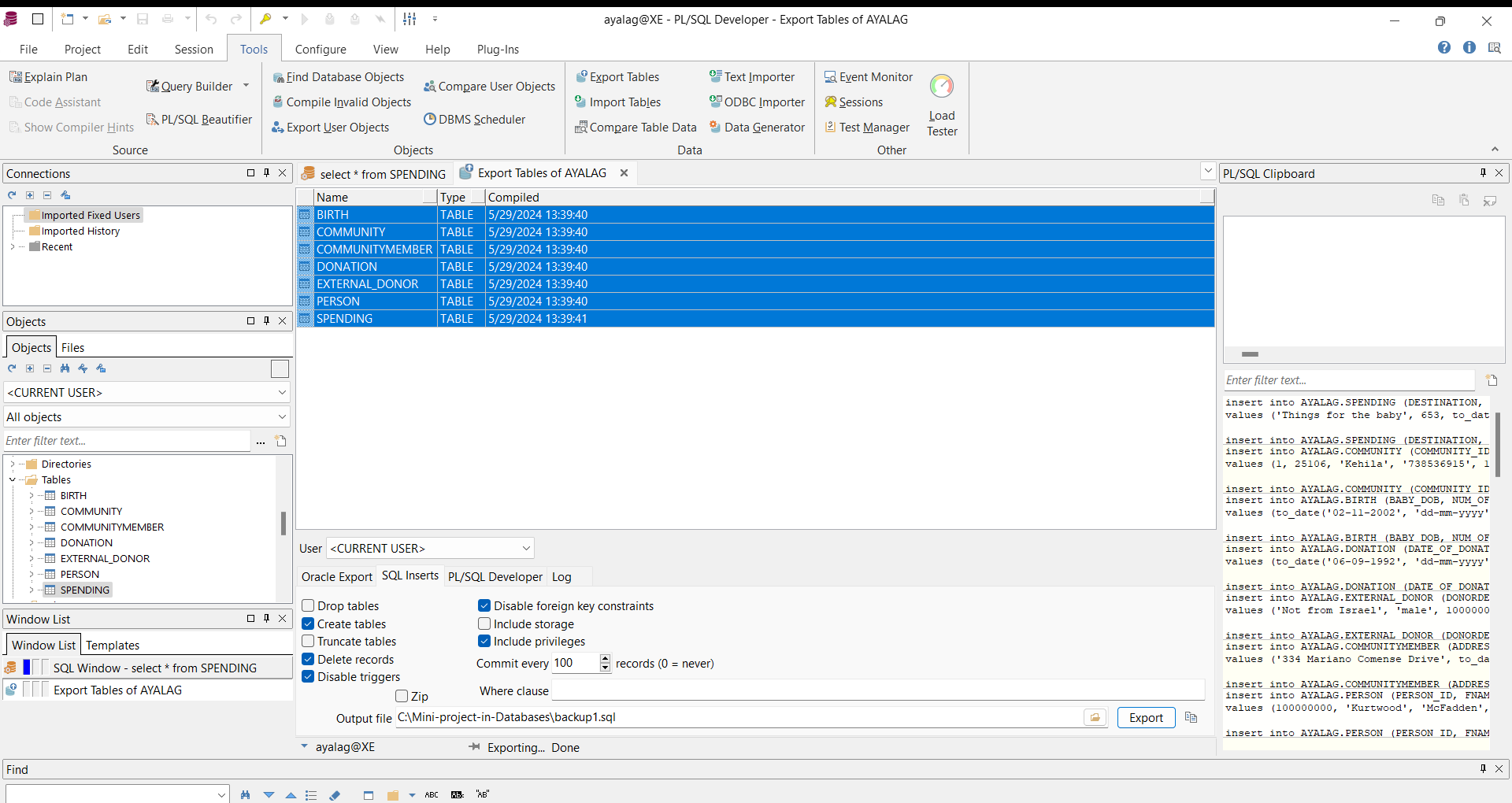
**Person:**

****

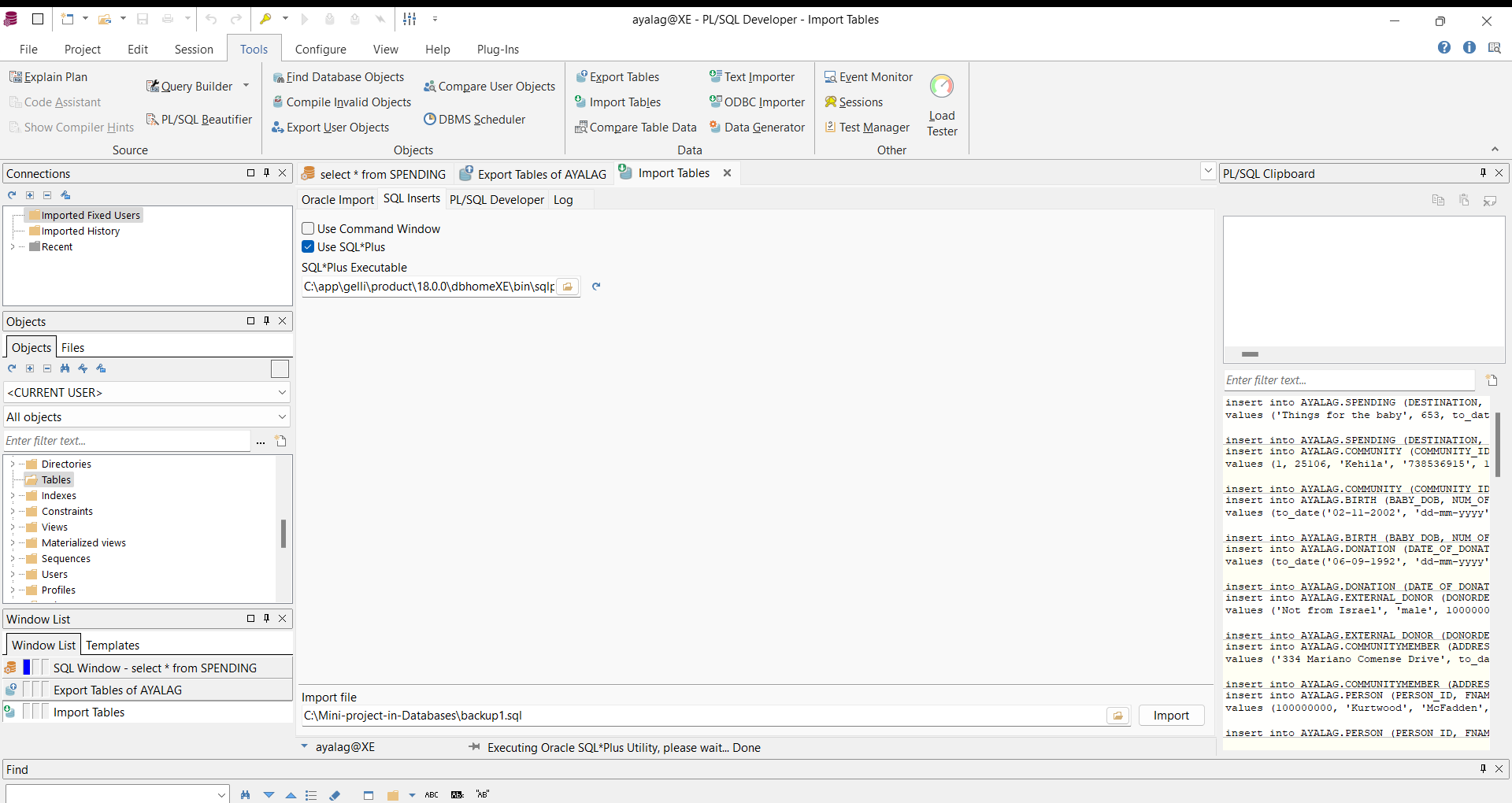
****

**גיבוי ושחזור הטבלאות:**

**גיבוי:**

****

**שחזור:**

****

**שינויים למערכת:**

**We removed the donation\_id column from the Community table.**

**alter table COMMUNITY drop column DONATION\_ID;**

**We added the community\_id column to the Donation table as a foreign key.**

**alter table DONATION add COMMUNITY\_ID INTEGER;**

**UPDATE DONATION**

**SET COMMUNITY\_ID = FLOOR(DBMS\_RANDOM.VALUE(1, 21));**

**COMMIT;**

**We removed the Spending table so that we can do the next step.**

**Drop Spending table**

**We minimized the Community table to 10 rows for easier management.**

**CREATE GLOBAL TEMPORARY TABLE top\_20\_community\_ids ON COMMIT PRESERVE ROWS AS**

**SELECT community\_id**

**FROM community**

**ORDER BY community\_id**

**FETCH FIRST 20 ROWS ONLY;**

**DELETE FROM community**

**WHERE community\_id NOT IN (SELECT community\_id FROM top\_20\_community\_ids);**

**DROP TABLE top\_20\_community\_ids;**

**We recreated the Spending table.**

**We added a community\_id column as a foreign key to the CommunityMember table.**

**alter table communitymember add COMMUNITY\_ID INTEGER;**

**We added a birth\_id column as a primary key to the Birth table.**

**alter table birth add birth\_id INTEGER;**

**CREATE SEQUENCE birth\_id\_seq START WITH 1;**

**UPDATE BIRTH SET birth\_id = birth\_id\_seq.NEXTVAL;**

**COMMIT;**

**BEGIN**

**EXECUTE IMMEDIATE 'ALTER TABLE BIRTH MODIFY (birth\_id PRIMARY KEY)';**

**END;**

**בגלל שרבו השינויים החלטנו לייצר את כל הטבלאות מחדש ולהלן הפקודות:**

CREATE TABLE person

(

person\_id INT NOT NULL,

fname VARCHAR2(30) NOT NULL,

lname VARCHAR2(30) NOT NULL,

phone\_number VARCHAR2(10) NOT NULL,

PRIMARY KEY (person\_id)

);

CREATE TABLE community

(

community\_id INT NOT NULL,

monthly\_donation NUMBER NOT NULL,

community\_type VARCHAR2(30) NOT NULL,

phone\_number VARCHAR2(10) NOT NULL,

PRIMARY KEY (community\_id)

);

CREATE TABLE communityMember

(

address VARCHAR2(30) NOT NULL,

date\_of\_birth DATE NOT NULL,

member\_id INT NOT NULL,

community\_id INT NOT NULL,

person\_id INT NOT NULL,

PRIMARY KEY (member\_id),

FOREIGN KEY (community\_id) REFERENCES community(community\_id),

FOREIGN KEY (person\_id) REFERENCES person(person\_id)

);

CREATE TABLE external\_donor

(

donorDescription VARCHAR2(300),

gender VARCHAR2(6) NOT NULL,

person\_id INT NOT NULL,

donor\_id INT NOT NULL,

PRIMARY KEY (donor\_id),

FOREIGN KEY (person\_id) REFERENCES person(person\_id)

);

CREATE TABLE donation

(

date\_of\_donation DATE NOT NULL,

donation\_amount NUMBER NOT NULL,

donation\_id INT NOT NULL,

person\_id INT NOT NULL,

community\_id INT NOT NULL,

FOREIGN KEY (community\_id) REFERENCES community(community\_id),

FOREIGN KEY (person\_id) REFERENCES person(person\_id)

);

CREATE TABLE birth

(

birth\_id INT NOT NULL,

baby\_DOB DATE NOT NULL,

num\_of\_children INT NOT NULL,

comments VARCHAR2(200) NOT NULL,

member\_id INT NOT NULL,

PRIMARY KEY (birth\_id),

FOREIGN KEY (member\_id) REFERENCES communityMember(member\_id)

);

CREATE TABLE spending

(

destination VARCHAR2(300) NOT NULL,

amount NUMBER NOT NULL,

spending\_date DATE NOT NULL,

spending\_id INT NOT NULL,

community\_id INT NOT NULL,

PRIMARY KEY (spending\_id),

FOREIGN KEY (community\_id) REFERENCES community(community\_id)

);

**לאור זאת זה גם מה שמכיל הקובץ של השינויים.**

**Constraints:**

ALTER TABLE community

ADD CONSTRAINT chk\_monthly\_donation\_positive

CHECK (monthly\_donation > 0);

ALTER TABLE person

MODIFY phone\_number DEFAULT '000000000';

ALTER TABLE external\_donor

MODIFY donorDescription VARCHAR2(300) NOT NULL;

A screenshot of a computer

Description automatically generated

**Queries:**

**Select**

רשימת חברות הקהילה שילדו ילד בשנה האחרונה, ממוינת לפי חודשים



**Select**

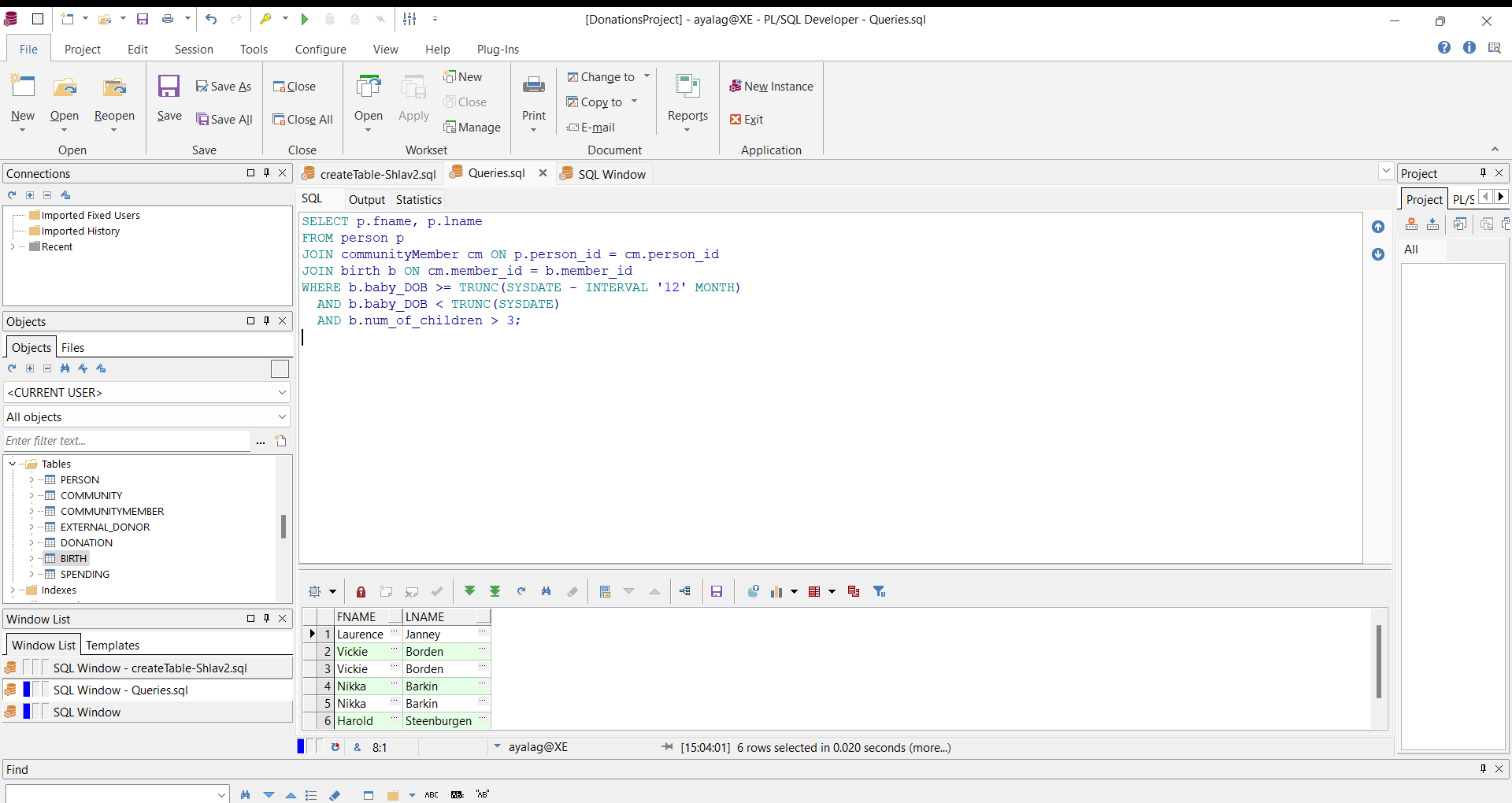
רשימת התורמים עם הפרטים שלהם (שם מלא ומספר טלפון) ואת הסכום הכולל של התרומות שהם תרמו לכל קהילה, ממוינת לפי קהילות ולאחר מכן בסדר יורד של סכום התרומות.

A screenshot of a computer

Description automatically generated

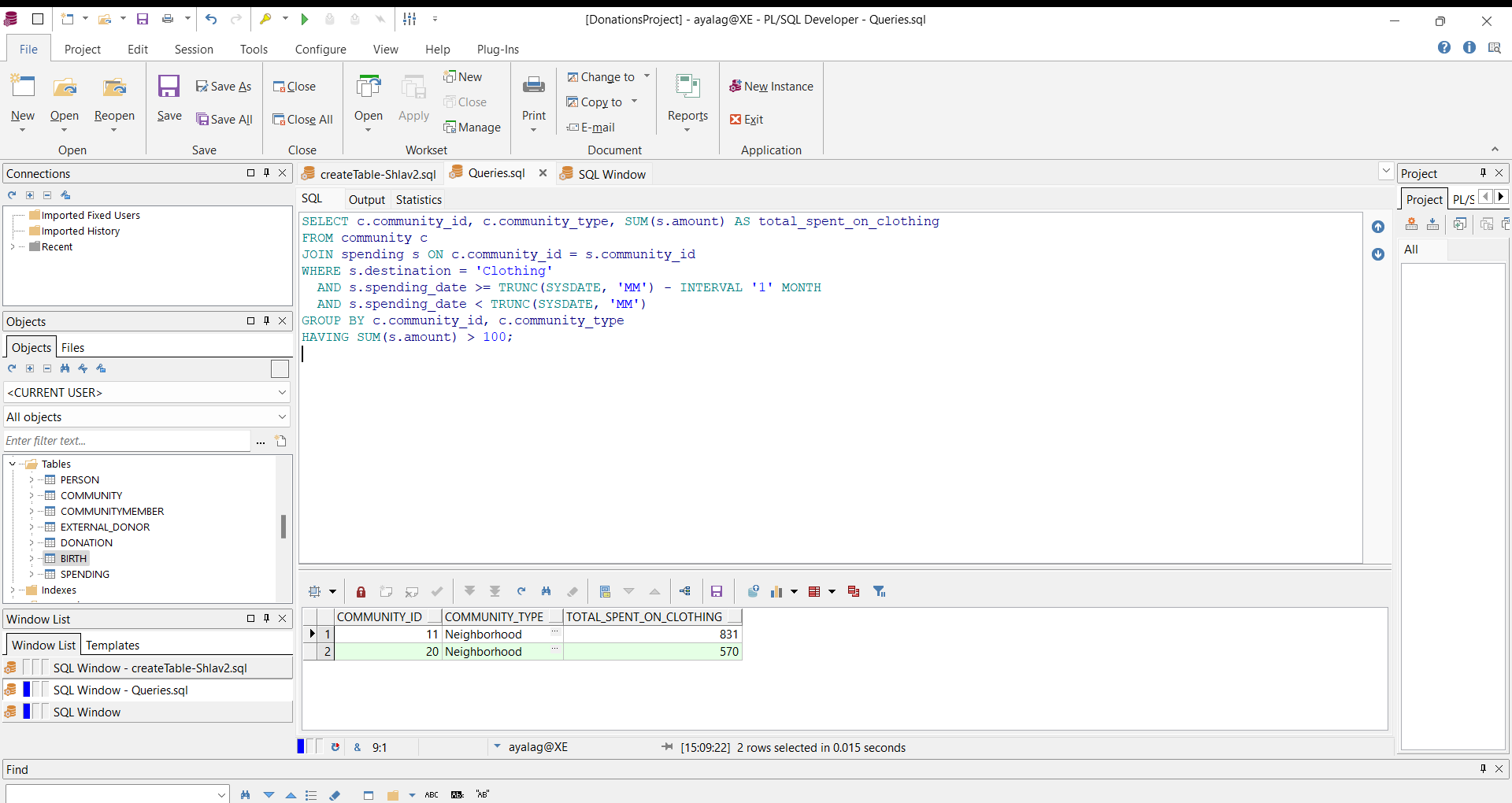
**Select**

רשימת חברי קהילה שילדו בשנה האחרונה ויש להם יותר משלשה ילדים.



**Select**

רשימת הקהילות שהוציאו בשנה האחרונה יותר מ 100 על הגדים. ממוינת לפי ת"ז וסוג קהילה.



**Update**

מעדכנים את התרומה החודשית של כל קהילה להיות כפול 5 במידה וסך התרומות שהשילה קחבלה בשנה האחרונה הינו פחות מ 100000.

**Before:**



**After:**

A screenshot of a computer

Description automatically generated

**Update**

מעדכנים את התיאור של כל התורמים החיצונים להיות אדם פרטי במידה והוא אי פעם תרם פחות מ-50000.

**Before:**

A screenshot of a computer

Description automatically generated

**After:**

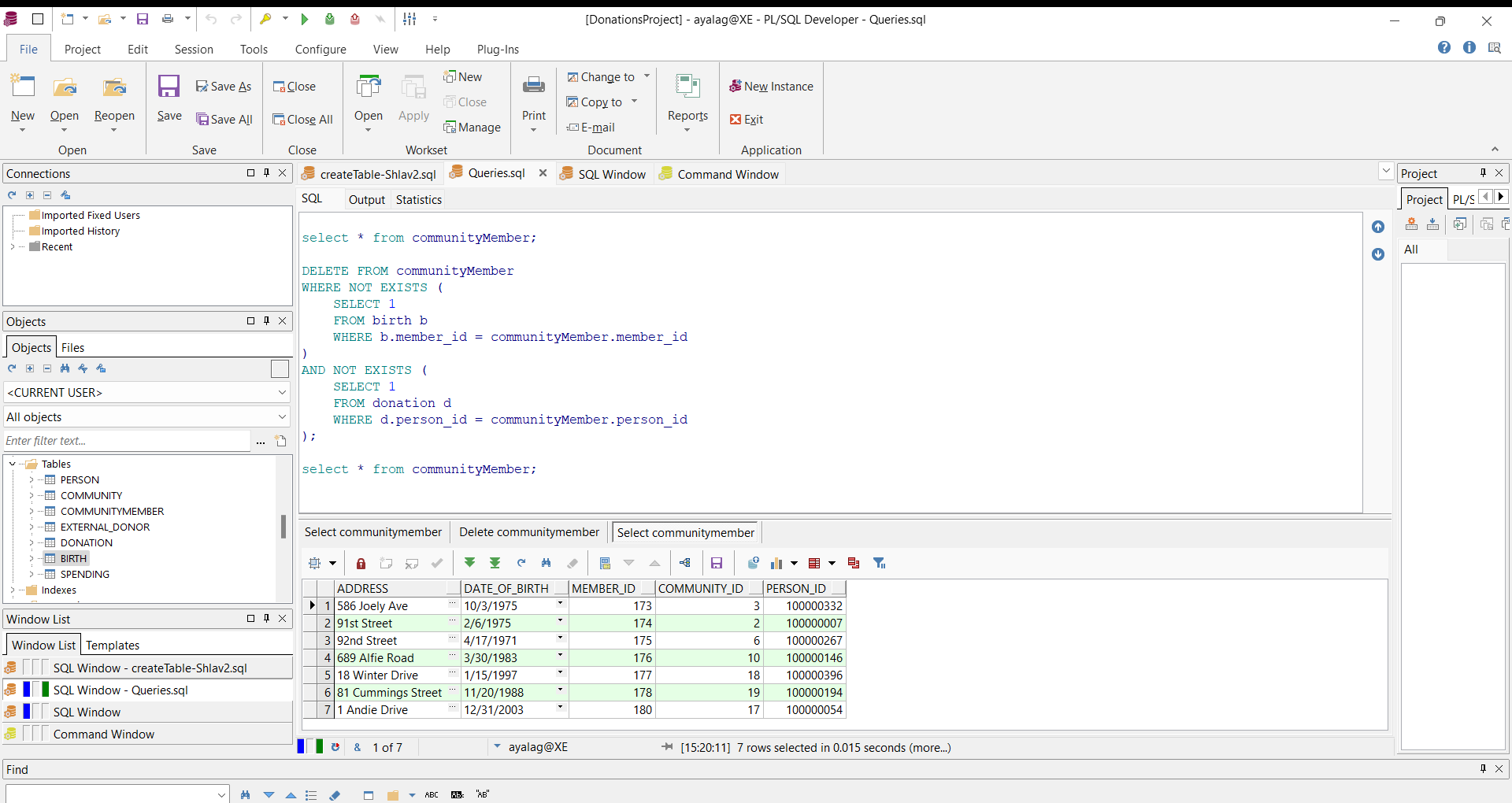
A screenshot of a computer

Description automatically generated

**Delete**

מוחקים את כל החברי קהילה שלעולם לא ילדו ולעולם לא תרמו.

**Before:**



**After:**

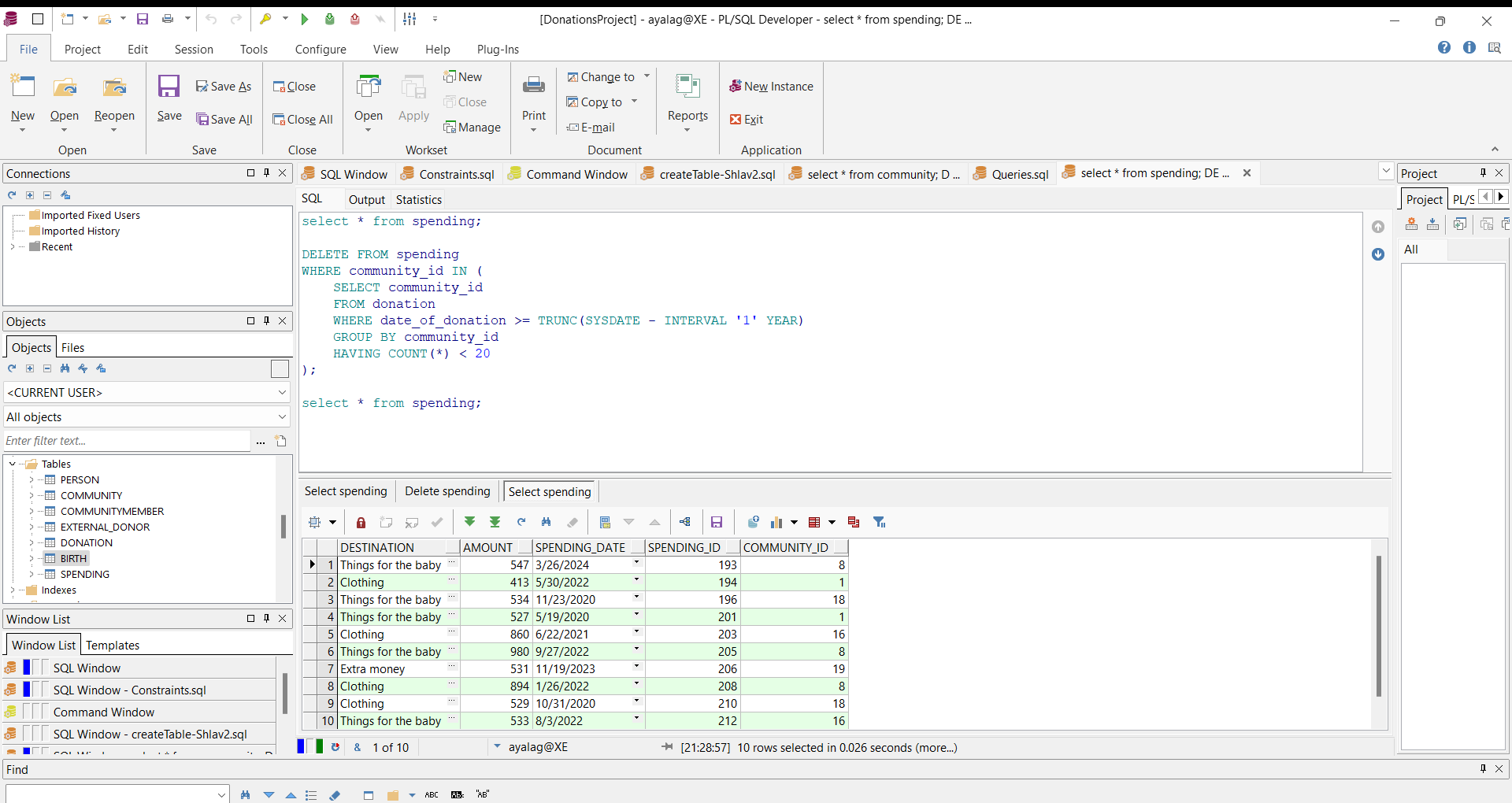
A screenshot of a computer

Description automatically generated

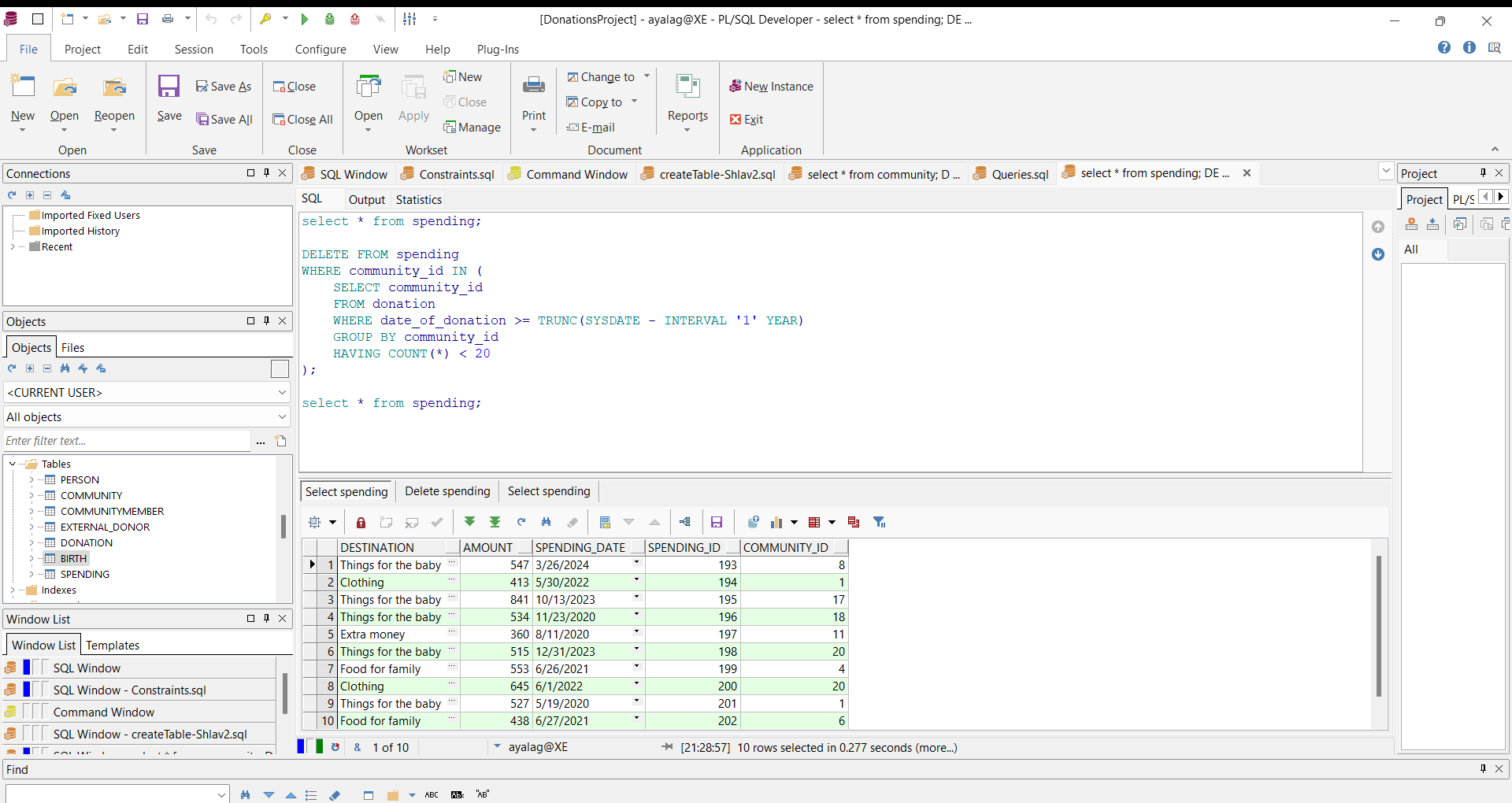
**Delete**

מוחקים את כל ההוצאות של קהילות שקיבלו פחות מ 20 תרומות בשנה האחרונה.

**Before:**



**After:**

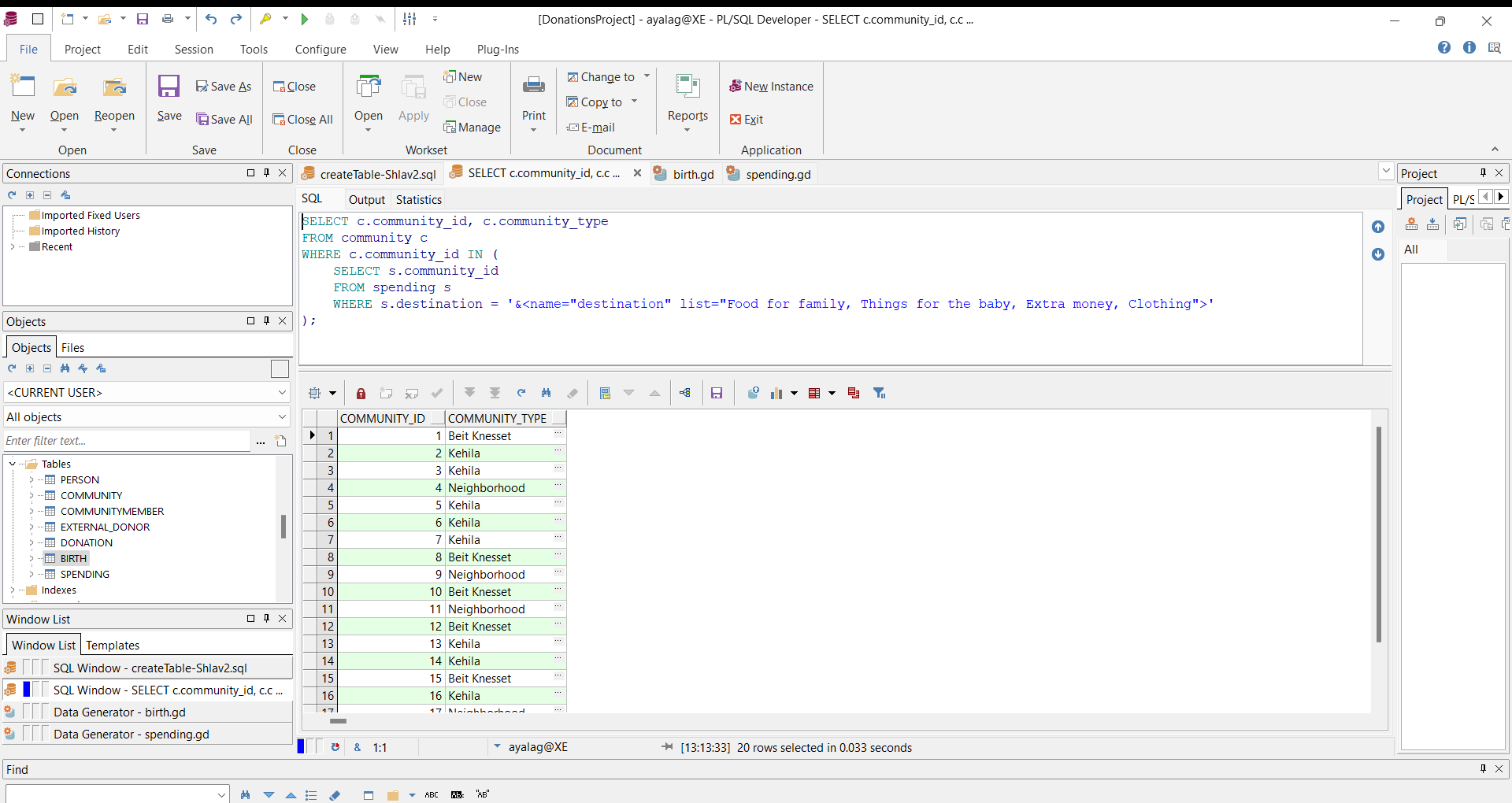


**Parameter Query**

בוחרים את כל החברי קהילה שילדו בטווח תאריכים מסוים.

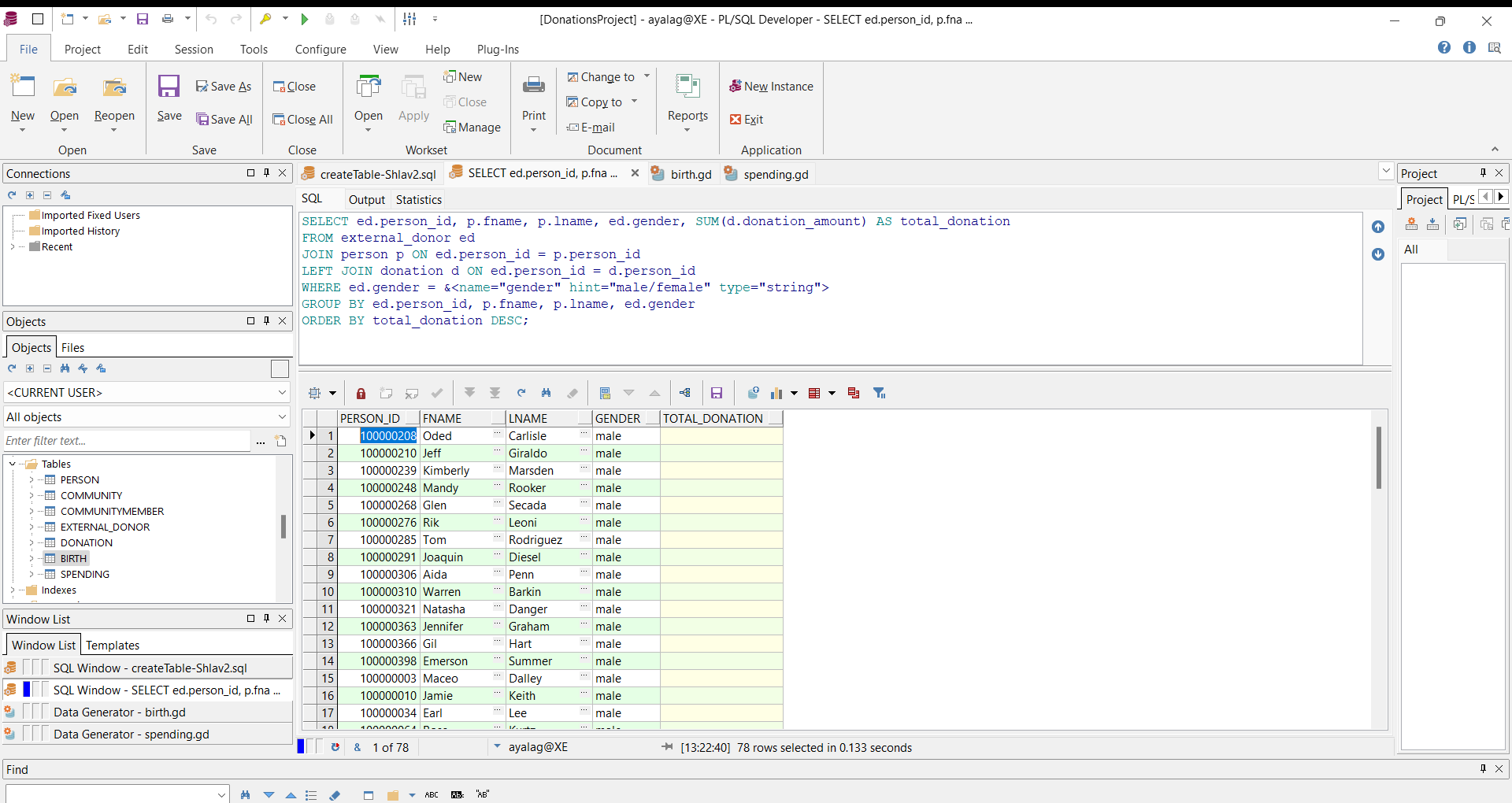


בוחרים את כל הקהילות שהוציאו כסף על דבר מסוים ( שאותו המשתמש בוחר)

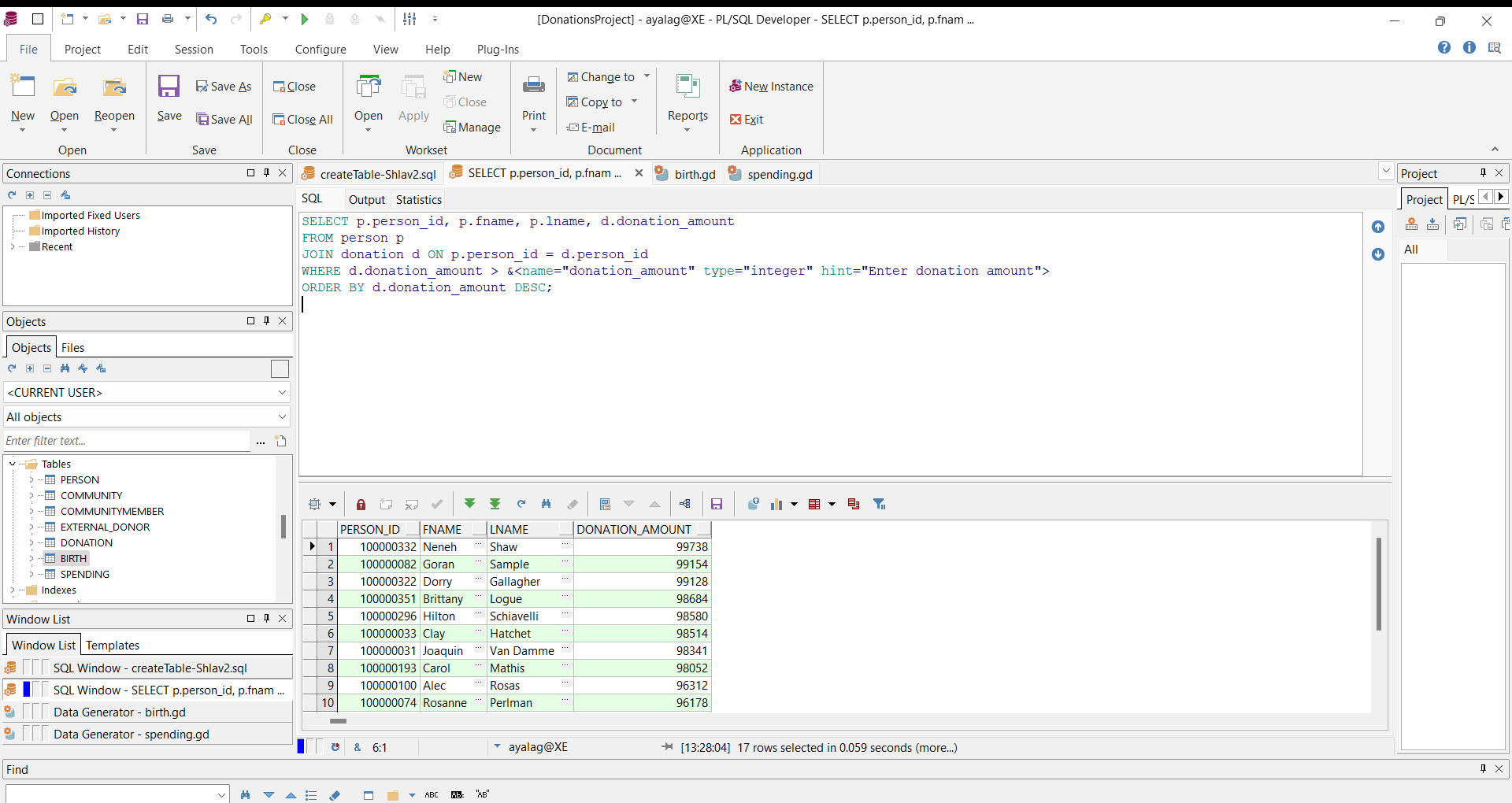


**Parameter Query**

בוחרים את כל התורמים החיצונים לפי מגדר וממינים לפי הסכום הכולל שהם תרמו.



בוחרים את כל האנשים שתרמו צעל לסכום מסוים. וממינים לפי הסכם הכולל שכל תורם תרם.



**Function**

הפונקציה הזאת מחשבת את הקהילה שהכי הרבה נשים ילדו בה בשנה האחרונה ומדפיסה את מספר הלידות. ובנוסף, הפונקציה מחשבת מהי הקהילה שהוציאה את הסכום הגבוה ביותר על אוכל ומדפיסה אותה וכמה היא הוציאה.

CREATE OR REPLACE NONEDITIONABLE FUNCTION find\_and\_print\_top\_community\_stats RETURN VARCHAR2

IS

v\_output VARCHAR2(1000); *-- Variable to store output for return*

v\_community\_id\_births NUMBER;

v\_community\_type\_births VARCHAR2(30);

v\_num\_of\_births NUMBER;

v\_community\_id\_food NUMBER;

v\_community\_type\_food VARCHAR2(30);

v\_amount\_spent\_on\_food NUMBER;

*-- Cursors declaration*

CURSOR c\_births IS

SELECT cm.community\_id,

c.community\_type,

COUNT(b.birth\_id) AS num\_of\_births

FROM birth b

JOIN communityMember cm ON b.member\_id = cm.member\_id

JOIN community c ON cm.community\_id = c.community\_id

WHERE b.baby\_DOB >= ADD\_MONTHS(TRUNC(SYSDATE, 'YEAR'), -36) *-- Last 3 years*

GROUP BY cm.community\_id, c.community\_type

ORDER BY COUNT(b.birth\_id) DESC

FETCH FIRST 1 ROWS ONLY;

CURSOR c\_food\_spending IS

SELECT c.community\_id,

c.community\_type,

SUM(s.amount) AS total\_spending\_on\_food

FROM spending s

JOIN community c ON s.community\_id = c.community\_id

WHERE s.destination = 'Food for family'

GROUP BY c.community\_id, c.community\_type

ORDER BY SUM(s.amount) DESC

FETCH FIRST 1 ROWS ONLY;

BEGIN

*-- Query to find community with the most births*

OPEN c\_births;

FETCH c\_births INTO v\_community\_id\_births, v\_community\_type\_births, v\_num\_of\_births;

CLOSE c\_births;

*-- Build output for community with most births*

v\_output := 'Community with Most Births:' || CHR(10);

v\_output := v\_output || ' Community ID: ' || v\_community\_id\_births || CHR(10);

v\_output := v\_output || ' Community Type: ' || v\_community\_type\_births || CHR(10);

v\_output := v\_output || ' Number of Births: ' || v\_num\_of\_births || CHR(10);

v\_output := v\_output || '----------------------------------' || CHR(10);

*-- Query to find community that spent the most amount on food*

OPEN c\_food\_spending;

FETCH c\_food\_spending INTO v\_community\_id\_food, v\_community\_type\_food, v\_amount\_spent\_on\_food;

CLOSE c\_food\_spending;

*-- Build output for community with most spending on food*

v\_output := v\_output || 'Community with Most Spending on Food:' || CHR(10);

v\_output := v\_output || ' Community ID: ' || v\_community\_id\_food || CHR(10);

v\_output := v\_output || ' Community Type: ' || v\_community\_type\_food || CHR(10);

v\_output := v\_output || ' Amount Spent on Food: ' || v\_amount\_spent\_on\_food || CHR(10);

v\_output := v\_output || '----------------------------------';

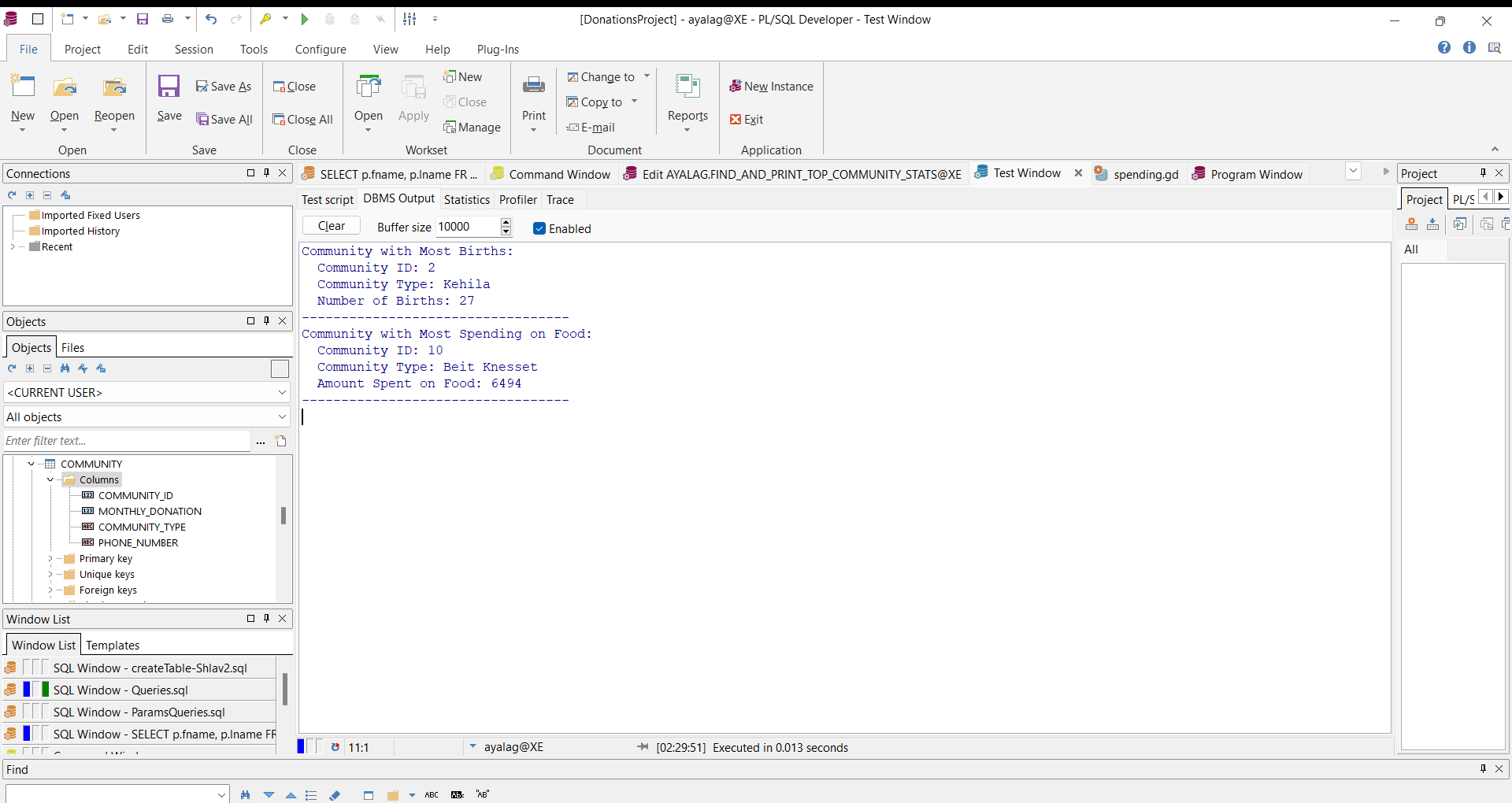
*-- Print output using DBMS\_OUTPUT*

DBMS\_OUTPUT.PUT\_LINE(v\_output);

*-- Return the output for potential further use*

RETURN v\_output;

END;



Procedure

**הפרוצדורה הזאת מחשבת עבור כל קהילה את מספר התרומות שהיא קיבלה בחמש שנים האחרונות וכמה כסף היא הוציאה בחמש השנים האחרונות על כל סוג של הוצאה ואת האחוז שהיא הוציאה על כל הוציאה מסך כל התרומות שהיא קיבלה בחמש שנים האחרונות.**

CREATE OR REPLACE PROCEDURE community\_donation\_spending\_summary IS

*-- Custom exception declaration*

no\_food\_spending\_exception EXCEPTION;

PRAGMA EXCEPTION\_INIT(no\_food\_spending\_exception, -20001); *-- Assigning error code*

v\_food\_spending\_found BOOLEAN := FALSE; *-- Flag to track if any food spending was found*

BEGIN

FOR rec IN (

SELECT c.community\_id,

c.community\_type,

SUM(d.donation\_amount) AS total\_donations,

SUM(CASE WHEN s.destination = 'Food for family' THEN s.amount ELSE 0 END) AS food\_spending,

SUM(CASE WHEN s.destination = 'Things for the baby' THEN s.amount ELSE 0 END) AS baby\_things\_spending,

SUM(CASE WHEN s.destination = 'Extra money' THEN s.amount ELSE 0 END) AS extra\_money,

SUM(CASE WHEN s.destination = 'Clothing' THEN s.amount ELSE 0 END) AS clothing\_spending,

ROUND(SUM(CASE WHEN s.destination = 'Food for family' THEN s.amount ELSE 0 END) / SUM(d.donation\_amount) \* 100, 2) AS food\_percentage,

ROUND(SUM(CASE WHEN s.destination = 'Things for the baby' THEN s.amount ELSE 0 END) / SUM(d.donation\_amount) \* 100, 2) AS baby\_things\_percentage,

ROUND(SUM(CASE WHEN s.destination = 'Extra money' THEN s.amount ELSE 0 END) / SUM(d.donation\_amount) \* 100, 2) AS extra\_money\_percentage,

ROUND(SUM(CASE WHEN s.destination = 'Clothing' THEN s.amount ELSE 0 END) / SUM(d.donation\_amount) \* 100, 2) AS clothing\_percentage

FROM community c

LEFT JOIN donation d ON c.community\_id = d.community\_id

LEFT JOIN spending s ON c.community\_id = s.community\_id

WHERE d.date\_of\_donation >= ADD\_MONTHS(TRUNC(SYSDATE, 'YYYY'), -60)

AND s.spending\_date >= ADD\_MONTHS(TRUNC(SYSDATE, 'YYYY'), -60)

GROUP BY c.community\_id, c.community\_type

ORDER BY c.community\_id

) LOOP

*-- Checking if food spending is zero*

IF rec.food\_spending = 0 THEN

v\_food\_spending\_found := TRUE; *-- Set flag to true if zero spending found*

END IF;

*-- Printing community details*

DBMS\_OUTPUT.PUT\_LINE('Community ID: ' || rec.community\_id);

DBMS\_OUTPUT.PUT\_LINE('Community Type: ' || rec.community\_type);

DBMS\_OUTPUT.PUT\_LINE('Total Donations: ' || NVL(rec.total\_donations, 0));

DBMS\_OUTPUT.PUT\_LINE('Food Spending: ' || NVL(rec.food\_spending, 0) || ' (' || NVL(rec.food\_percentage, 0) || '%)');

DBMS\_OUTPUT.PUT\_LINE('Baby Things Spending: ' || NVL(rec.baby\_things\_spending, 0) || ' (' || NVL(rec.baby\_things\_percentage, 0) || '%)');

DBMS\_OUTPUT.PUT\_LINE('Extra Money Spending: ' || NVL(rec.extra\_money, 0) || ' (' || NVL(rec.extra\_money\_percentage, 0) || '%)');

DBMS\_OUTPUT.PUT\_LINE('Clothing Spending: ' || NVL(rec.clothing\_spending, 0) || ' (' || NVL(rec.clothing\_percentage, 0) || '%)');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------');

END LOOP;

*-- Raise exception if no food spending was found*

IF NOT v\_food\_spending\_found THEN

RAISE\_APPLICATION\_ERROR(-20001, 'No money was spent on food for any community in the last year.');

END IF;

EXCEPTION

WHEN no\_food\_spending\_exception THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END community\_donation\_spending\_summary;

A screenshot of a computer

Description automatically generated

Program that runs the above function and procedure

BEGIN

*--:v\_result := find\_and\_print\_top\_community\_stats();*

community\_donation\_spending\_summary;

END;

Function

**הפונקציה הזאת מחשבת את סכום התרומות של תורם לשנה כלשהי.**

CREATE OR REPLACE FUNCTION get\_last\_year\_donations(person\_id IN NUMBER) RETURN NUMBER IS

last\_year\_donations NUMBER;

BEGIN

SELECT NVL(SUM(donation\_amount), 0)

INTO last\_year\_donations

FROM donation

WHERE person\_id = person\_id

AND date\_of\_donation >= ADD\_MONTHS(SYSDATE, -12);

RETURN last\_year\_donations;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

RAISE;

END get\_last\_year\_donations;

Procedure

**הפרוצדורה analyze\_donations בודקת את את התרומות של כל תורם, לפי השינויים בתרומות היא מעדכנת את ההערות על כל תורם, ומסווגת אותו לפי הסכום שתרם בסוף, היא יוצרת רשימה עם כל התוצאות, הכוללת מידע על התורמים והתרומות שלהם.**

CREATE OR REPLACE PROCEDURE analyze\_donations (out\_donors OUT SYS\_REFCURSOR) IS

CURSOR donor\_cursor IS

SELECT p.person\_id, p.fname, p.lname

FROM person p;

donor\_id person.person\_id%TYPE;

fname person.fname%TYPE;

lname person.lname%TYPE;

last\_year\_donations NUMBER;

previous\_years\_avg NUMBER;

change\_in\_donations NUMBER;

comments VARCHAR2(100);

classification VARCHAR2(10);

TYPE donor\_record\_type IS RECORD (

donor\_id person.person\_id%TYPE,

fname person.fname%TYPE,

lname person.lname%TYPE,

last\_year\_donations NUMBER,

change\_in\_donations NUMBER,

comments VARCHAR2(100),

classification VARCHAR2(10)

);

donor\_record donor\_record\_type;

BEGIN

OPEN donor\_cursor;

LOOP

FETCH donor\_cursor INTO donor\_id, fname, lname;

EXIT WHEN donor\_cursor%NOTFOUND;

*-- קבלת סכום התרומות של השנה האחרונה*

last\_year\_donations := get\_last\_year\_donations(donor\_id);

*-- קבלת ממוצע התרומות של השנים הקודמות*

BEGIN

SELECT NVL(AVG(donation\_amount), 0)

INTO previous\_years\_avg

FROM donation

WHERE person\_id = donor\_id

AND date\_of\_donation < ADD\_MONTHS(SYSDATE, -12);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

previous\_years\_avg := 0;

WHEN OTHERS THEN

RAISE;

END;

*-- חישוב השינוי בתרומות*

change\_in\_donations := last\_year\_donations - previous\_years\_avg;

*-- קביעת ההערות*

IF ABS(change\_in\_donations) > 1000 THEN

comments := 'Need to update';

ELSE

comments := 'No update needed';

END IF;

*-- סיווג התורם*

IF last\_year\_donations >= 5000 THEN

classification := 'VIP';

ELSE

classification := 'Regular';

END IF;

*-- עדכון הערות וסיווג בתורמים*

UPDATE external\_donor

SET donorDescription = comments

WHERE person\_id = donor\_id;

*-- הכנת רשומה להצגה*

donor\_record.donor\_id := donor\_id;

donor\_record.fname := fname;

donor\_record.lname := lname;

donor\_record.last\_year\_donations := last\_year\_donations;

donor\_record.change\_in\_donations := change\_in\_donations;

donor\_record.comments := comments;

donor\_record.classification := classification;

*-- הצגת התוצאות*

DBMS\_OUTPUT.PUT\_LINE('Donor ID: ' || donor\_record.donor\_id || ', Last Year Donations: ' || donor\_record.last\_year\_donations || ', Change: ' || donor\_record.change\_in\_donations || ', Comments: ' || donor\_record.comments || ', Classification: ' || donor\_record.classification);

END LOOP;

CLOSE donor\_cursor;

*-- פתיחת Ref Cursor להחזרת התוצאות*

OPEN out\_donors FOR

SELECT p.person\_id, p.fname, p.lname, e.donorDescription, e.gender, c.monthly\_donation

FROM person p

JOIN donation d ON p.person\_id = d.person\_id

JOIN community c ON d.community\_id = c.community\_id

JOIN external\_donor e ON p.person\_id = e.person\_id;

END analyze\_donations;

Program that runs the above function and procedure

**התוכנית - מנתחת תרומות של תורמים, מעדכנת הערות וסיווג רמה ומציגה פרטים של תורמים.**

DECLARE

out\_donors SYS\_REFCURSOR;

donor\_id person.person\_id%TYPE;

fname person.fname%TYPE;

lname person.lname%TYPE;

donorDescription external\_donor.donorDescription%TYPE;

gender external\_donor.gender%TYPE;

monthly\_donation community.monthly\_donation%TYPE;

BEGIN

*-- הגדלת גודל הבופר של DBMS\_OUTPUT*

DBMS\_OUTPUT.ENABLE(1000000);

*-- קריאה לפרוצדורה לניתוח הנתונים*

analyze\_donations(out\_donors);

*-- קריאה לתוצאות מתוך ה-Ref Cursor*

LOOP

FETCH out\_donors INTO donor\_id, fname, lname, donorDescription, gender, monthly\_donation;

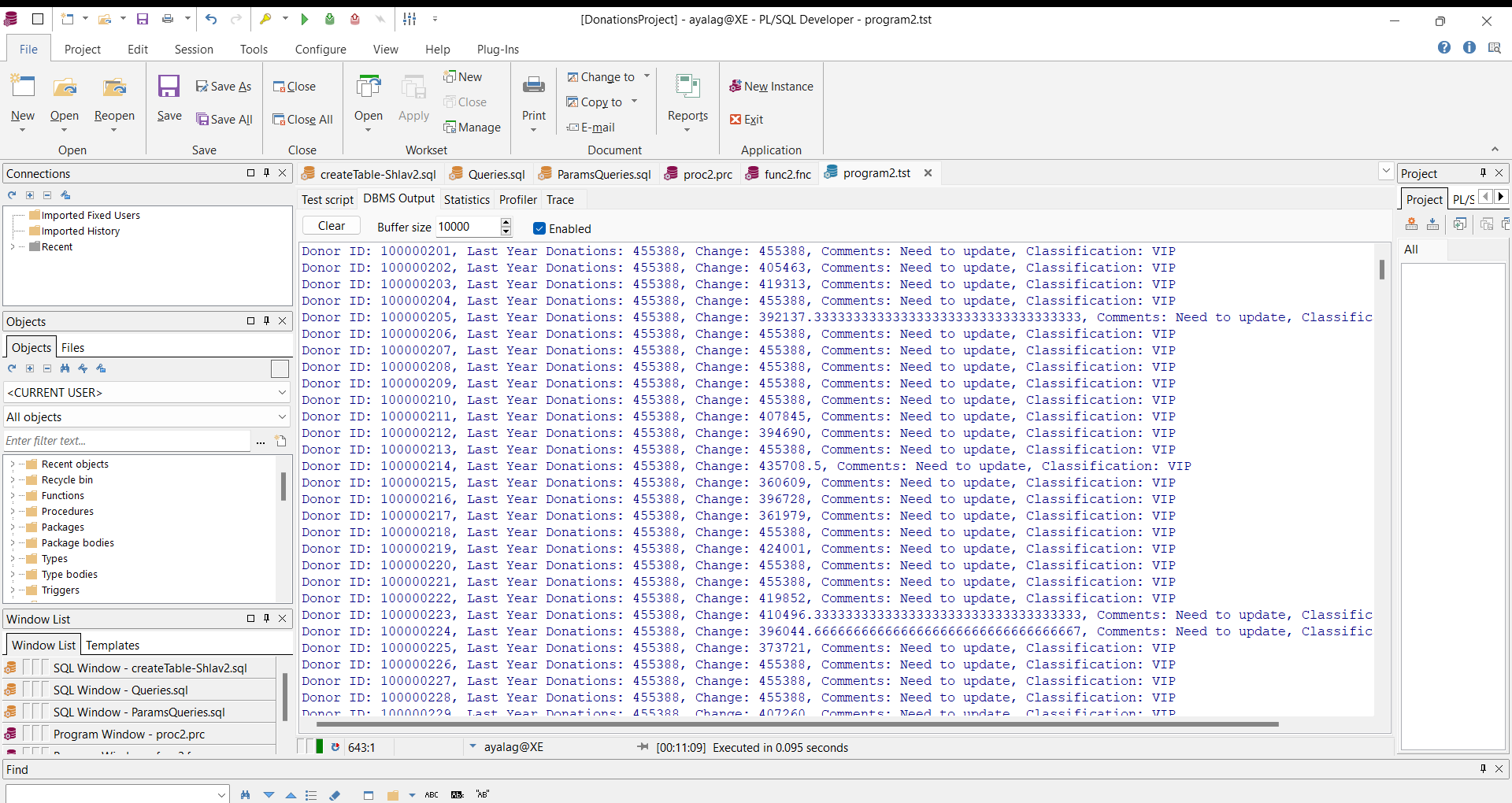
EXIT WHEN out\_donors%NOTFOUND;

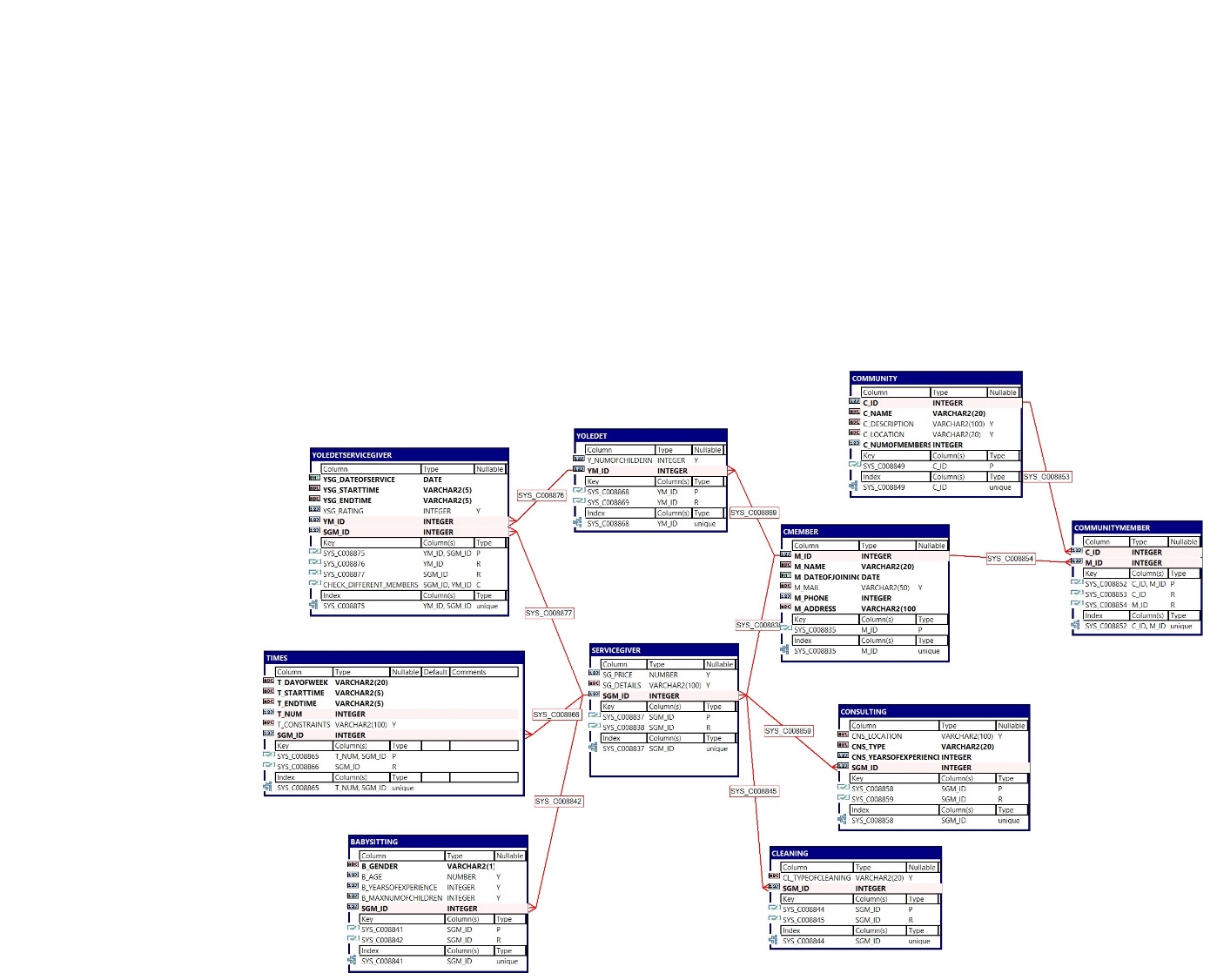
DBMS\_OUTPUT.PUT\_LINE('Donor ID: ' || donor\_id || ', Name: ' || fname || ' ' || lname || ', Description: ' || donorDescription || ', Gender: ' || gender || ', Monthly Donation: ' || monthly\_donation);

END LOOP;

CLOSE out\_donors;

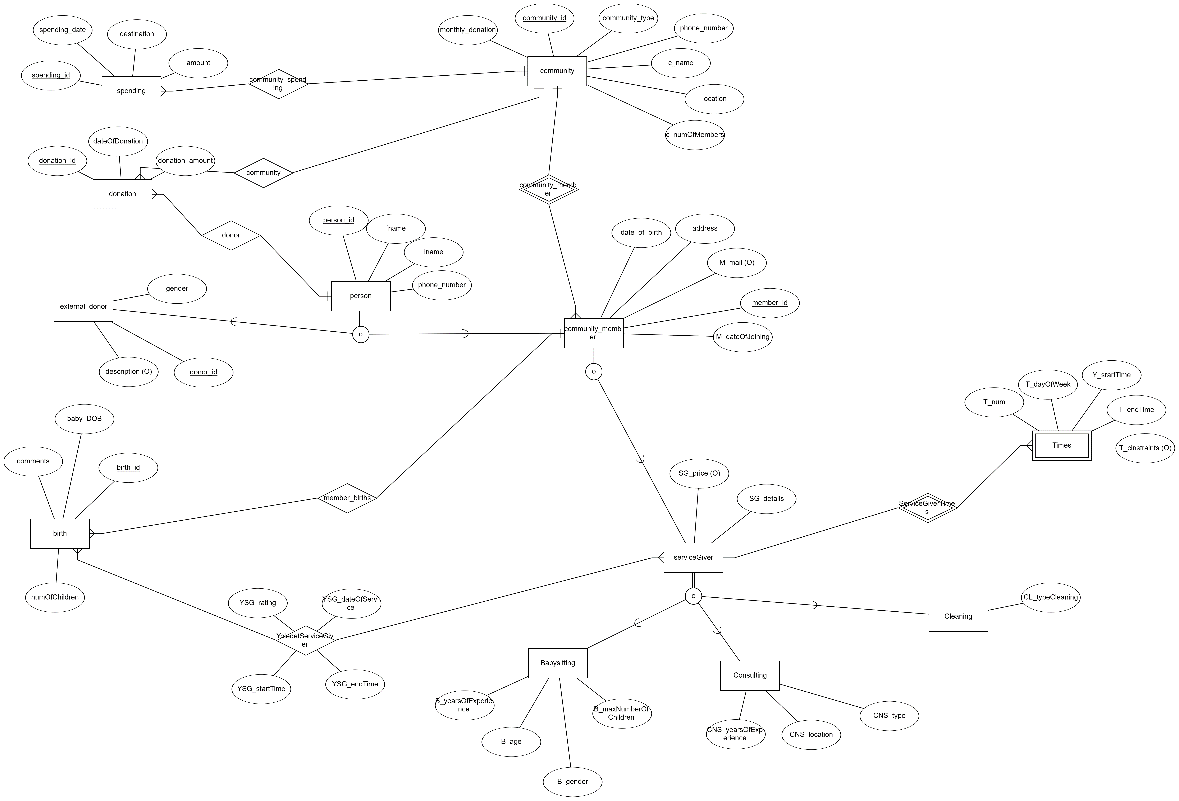
END;



1.      DSD של האגף החדש

2.      ERD אגף חדש

3.      ERD משותף



4.      DSD לאחר אינטגרציה

5.      פקודות שינוי ויצירה של טבלאות בקובץ Integrate.sql

6.      פקודות ליצירת המבטים והשאילתות על המבטים בשם Views.sql

7.      קובץ גיבוי מעודכן בשם backup4