ח' בסיון ה'תשפ"ב

גירסה 1.1.4

צור איתן לוי 205431935  
אליסף דימנט 204006415

מיני פרויקט בבסיסי נתונים

***גמ"ח כספים***

תוכן

[שלב א 3](#_Toc105524655)

[תיאור המערכת 3](#_Toc105524656)

[ישויות בסיס הנתונים: 3](#_Toc105524657)

[תיאור מילולי של טבלאות המערכת: 3](#_Toc105524658)

[דיאגרמת ERD: 4](#_Toc105524659)

[תרשים DSD: 5](#_Toc105524660)

[מילון מונחים: 6](#_Toc105524661)

[יצירת הטבלאות (Create Table): 7](#_Toc105524662)

[הזנת נתונים לטבלאות והצגתם (insert and select): 10](#_Toc105524663)

[עדכון הטבלאות (update): 15](#_Toc105524664)

[מחיקת רשומות (delete): 17](#_Toc105524665)

[מחיקת עמודות / טבלאות (drop): 19](#_Toc105524666)

[שלב ב 22](#_Toc105524667)

[יצירת ישויות (שורות) בטבלאות: 22](#_Toc105524668)

[יצירת קבצי csv ע"י Mockaroo: 22](#_Toc105524669)

[טעינת קבצי csv ל plsql: 23](#_Toc105524670)

[יצירת ישויות ע"י (plsql) data generator: 27](#_Toc105524671)

[גיבוי ואחזור הנתונים: 41](#_Toc105524672)

[גיבוי הנתונים: 41](#_Toc105524673)

[אחזור המידע: 42](#_Toc105524674)

[**שאילתות SQL**: 45](#_Toc105524675)

[אינדקסינג: 53](#_Toc105524676)

[טבלת השוואה לפני – אחרי האינדקסים: 53](#_Toc105524677)

[אינדקסים יעילים: 54](#_Toc105524678)

[אינדקסים לא-יעילים: 58](#_Toc105524679)

[שלב ג 60](#_Toc105524680)

[אינטגרציה: 60](#_Toc105524681)

[שלב ד 61](#_Toc105524682)

[Views: 61](#_Toc105524683)

[View 1: 61](#_Toc105524684)

[View 2: 64](#_Toc105524685)

[Procedures 67](#_Toc105524686)

[Procedure 1: 67](#_Toc105524687)

[Procedure 2: 69](#_Toc105524688)

[Functions: 71](#_Toc105524689)

[Functions 1: 71](#_Toc105524690)

[Function 2: 73](#_Toc105524691)

[Triggers: 75](#_Toc105524692)

[Triggers 1: 75](#_Toc105524693)

[Trigger 2: 77](#_Toc105524694)

[Git: 80](#_Toc105524695)

# שלב א

## תיאור המערכת

בפרויקט זה בחרנו לממש בסיס נתונים עבור גמ"ח כספים באופן גנרי.

גמ"ח הכספים בנוי באופן הבא:

אדם שמעוניין לקבל סכום כסף מהגמ"ח (להלן **"הלווה"**) צריך לפנות לגמ"ח הכספים (להלן "**המלווה"**), המלווה קובע עם הלווה את תנאי ההלוואה – סכום, תשלומים, מועד תחילת הפירעון, מועד סיום הפירעון ואמצעי התשלום.

בנוסף על הלווה להחתים לפחות **ערב** אחד על מסמך ההלוואה.

המלווה יאמת את פרטי הבנק של הלווה ולאחר מכן יבקש אישור וחתימה **מנשיא הגמ"ח**, ולאחר אישור סופי זה הגמ"ח ינפיק צ'ק בסכום שסוכם עבור הלווה.

## ישויות בסיס הנתונים:

* מלווה – Lender
* הלוואה – Loan
* תשלום – Payment
* חשבון הבנק – BankAccount
* בנק – Bank
* אדם – Person
* נשיא הגמ"ח – President
* ערב – Guarantee
* לווה – Loaner

## תיאור מילולי של טבלאות המערכת:

Lender (lenderBN, lenderName, lenderAddress, lenderPhone, LenderMail)

Loan (loanID, payCode, loanDate, PID, lenderBN)

Payment (payCode, dueDate, startDate, totalAmount, methodOfPayment, PID)

BankAccount (accountNumber, bankBN, balance, PID)

Bank (bankBN, bankName)

Person (PID, PfullName, Paddress, Pphone, Pmail)

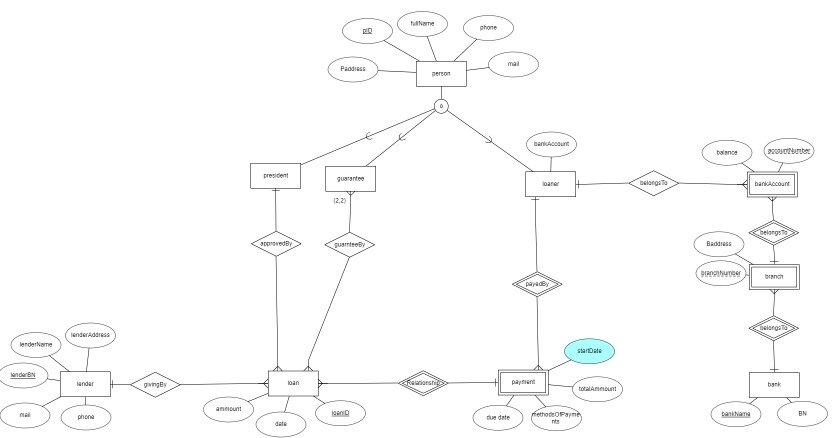
President (PID)

Guarantee (PID)

Loaner(PID, bankAccount)

GuarnteeBy (PID, loanID, payCode)

## דיאגרמת ERD:

**

## תרשים DSD:

## מילון מונחים:

|  |  |  |
| --- | --- | --- |
| Table name | Attribute | Explenation |
| Lender | lenderName | שם המלווה |
| lenderAddress | כתובת המלווה |
| lenderPhone | טלפון המלווה |
| LenderMail | כתובת מייל של המלווה |
| lenderBN | מספר העסק של הגמ"ח |
|  | |
| Person | PfullName | שם מלא |
| PID | תעודת זהות |
| PAddress | כתובת |
| Pphone | טלפון |
| Pmail | כתובת מייל |
|  | |
| Persident | PID | תעודת זהות של הנשיא |
|  | |
| Guarantee | PID | תעודת זהות של הערב |
|  | |
| Loaner | PID | תעודת זהות של הלווה |
| BankAccount | מספר חשבון בנק של הלווה |
|  | |
| GuaranteeBy | PID | תעודת זהות של הערב |
| LoanID | מספר מזהה להלוואה |
| payCode | מספר מזהה לעסקה |
|  | |
| Bank | BankBN | מספר מזהה של הבנק (ח.פ) |
| BankName | שם הבנק |
|  | |
| BankAccount | AccountNumber | מספר חשבון הבנק |
| BankBN | מספר מזהה של הבנק |
| balance | יתרה בחשבון |
| PID | תעודת זהות של בעל החשבון |
|  | |
| payment | payCode | מספר מזהה לעסקה |
| dueDate | תאריך פירעון סופי להלוואה |
| startDate | תחילת תשלום ההלוואה |
| totalAmount | סכום ההלוואה סך הכל |
| methodOfPayment | שיטת תשלום (צ'ק, אשראי, מזומן) |
| Pid | תעודת זהות של הלווה |
|  |  |  |
| Loan | loanID | מספר מזהה להלוואה |
| payCode | מספר מזהה לעסקה |
| loanDate | תאריך קבלת ההלוואה |
| PID | תעודת זהות של הנשיא |
| LenderBN | מספר מזהה של הגמ"ח |

## יצירת הטבלאות (Create Table):

Lender Table:

CREATE TABLE lender

(

lenderName VARCHAR(40) NOT NULL,

lenderAddress VARCHAR(40) NOT NULL,

lenderPhone VARCHAR(40) NOT NULL,

LenderMail VARCHAR(40) NOT NULL,

lenderBN INT NOT NULL,

PRIMARY KEY (lenderBN)

);

Person Table:

CREATE TABLE person

(

PfullName VARCHAR(40) NOT NULL,

PID INT NOT NULL,

Paddress VARCHAR(40) NOT NULL,

pPhone VARCHAR(40) NOT NULL,

Pmail VARCHAR(40) NOT NULL,

PRIMARY KEY (PID)

);

Loaner Table:

CREATE TABLE loaner

(

bankAccount INT NOT NULL,

PID INT NOT NULL,

PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES person(PID)

);

President Table:

CREATE TABLE president

(

PID INT NOT NULL,

PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES person(PID)

);

Guarantee Table:

CREATE TABLE guarantee

(

PID INT NOT NULL,

PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES person(PID)

);

Bank Table:

CREATE TABLE bank

(

bankName VARCHAR(40) NOT NULL,

bankBN INT NOT NULL,

PRIMARY KEY (bankBN)

);

Payment Table:

CREATE TABLE payment

(

dueDate DATE NOT NULL,

startDate DATE NOT NULL,

totalAmount INT NOT NULL,

methodsOfPayments VARCHAR(40) NOT NULL,

payCode INT NOT NULL,

PID INT NOT NULL,

PRIMARY KEY (payCode),

FOREIGN KEY (PID) REFERENCES loaner(PID)

);

Loan Table:

CREATE TABLE loan

(

loanDate DATE NOT NULL,

loanID INT NOT NULL,

payCode INT NOT NULL,

PID INT NOT NULL,

lenderBN INT NOT NULL,

PRIMARY KEY (loanID, payCode),

FOREIGN KEY (payCode) REFERENCES payment(payCode),

FOREIGN KEY (PID) REFERENCES president(PID),

FOREIGN KEY (lenderBN) REFERENCES lender(lenderBN)

);

Bank Account Table:

CREATE TABLE bankAccount

(

accountNumber INT NOT NULL,

balance INT NOT NULL,

bankBN INT NOT NULL,

PID INT NOT NULL,

PRIMARY KEY (accountNumber, bankBN),

FOREIGN KEY (bankBN) REFERENCES bank(bankBN),

FOREIGN KEY (PID) REFERENCES loaner(PID)

);

GuarnteeBy Table:

CREATE TABLE guarnteeBy

(

PID INT NOT NULL,

loanID INT NOT NULL,

payCode INT NOT NULL,

PRIMARY KEY (PID, loanID, payCode),

FOREIGN KEY (PID) REFERENCES guarantee(PID),

FOREIGN KEY (loanID, payCode) REFERENCES loan(loanID, payCode)

);

## הזנת נתונים לטבלאות והצגתם (insert and select):

Lender Table:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Person Table:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

President Table:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Guarantee Table:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Bank Table:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Loaner Table:

תמונה שמכילה טקסט

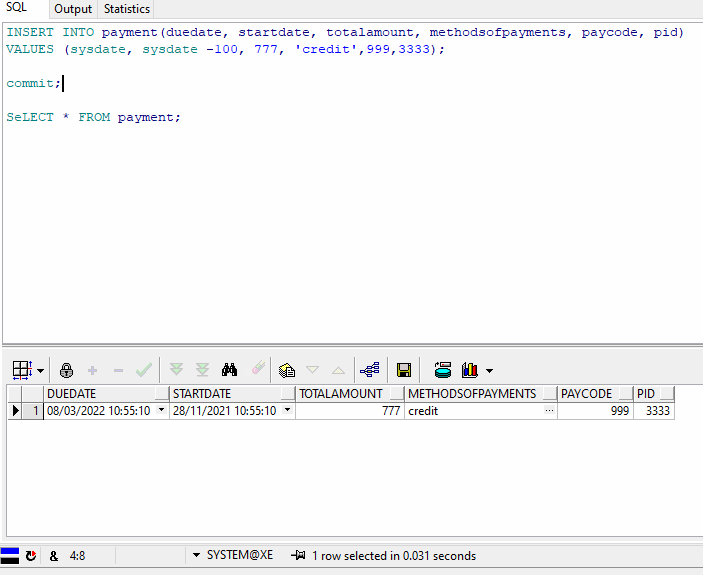
התיאור נוצר באופן אוטומטי

Bank Account Table:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Payment Table:



Loan Table:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

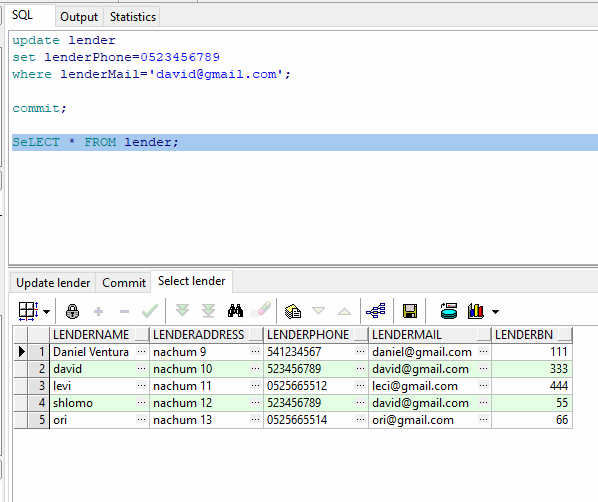
Guarantee By Table:

תמונה שמכילה טקסט

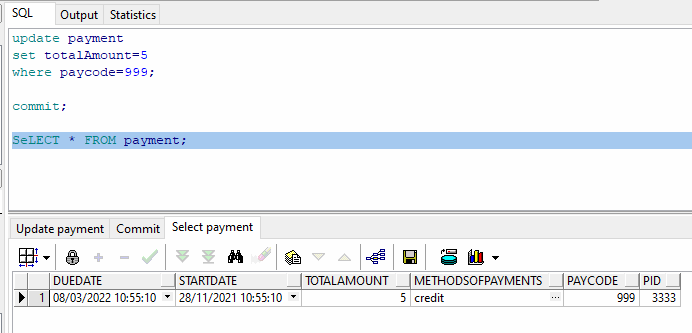
התיאור נוצר באופן אוטומטי

## עדכון הטבלאות (update):

Lender Table:



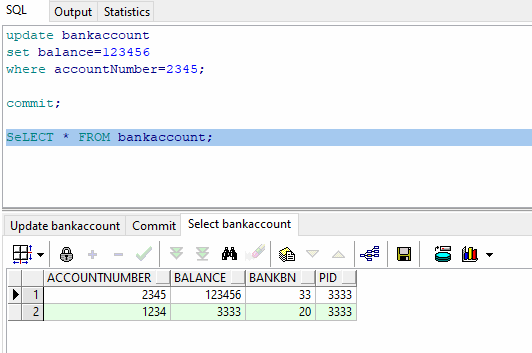
Payment Table:



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Bank Account Table:



## מחיקת רשומות (delete):

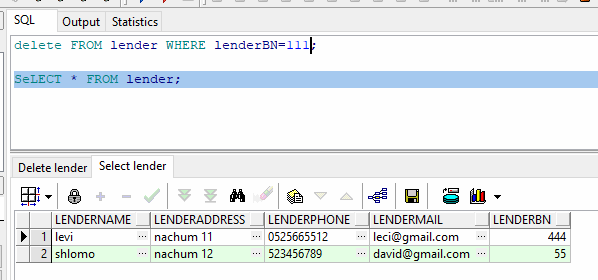
Lender Table:

תמונה שמכילה טקסט

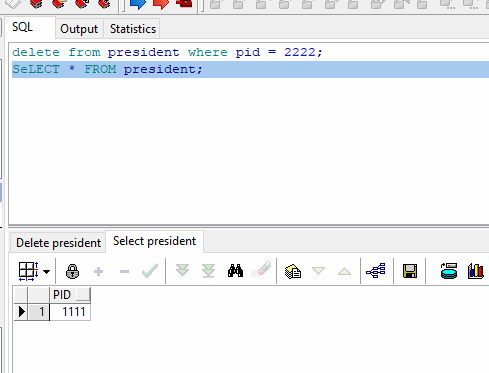
התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

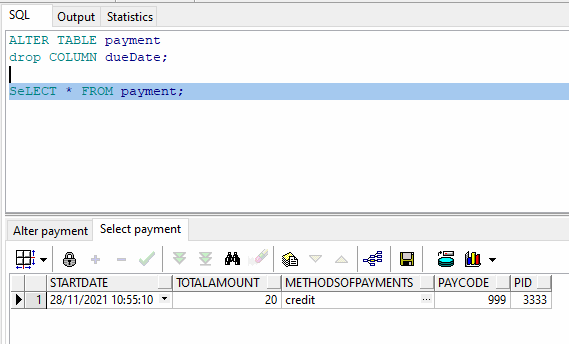


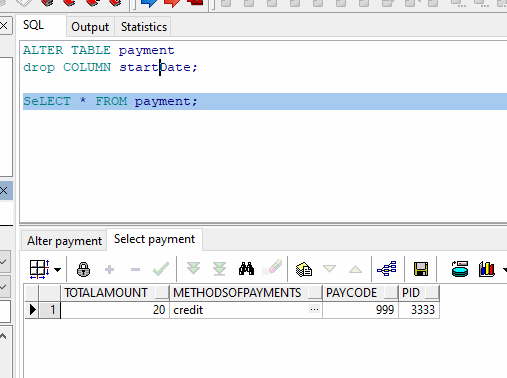
President Table:



## מחיקת עמודות / טבלאות (drop):

Payment Table:





כשניסינו למחוק את payment table קיבלנו הודעת שגיאה שלא ניתן למחוק את הטבלה כיון שיש לה בנים לכן נדרשנו להשתמש בפקודה "cascade" בכדי לבצע מחיקה "עמוקה".

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

ואכן ניתן להסיק שהטבלה נחקה כיון שבפקודת select לא ניתן למצוא אותה "does not exist"

Person Table:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

# שלב ב

## יצירת ישויות (שורות) בטבלאות:

### יצירת קבצי csv ע"י Mockaroo:

עבור ישות הבנק נחולל מידע עבור שתי השדות שלו – bankBN, bankName.

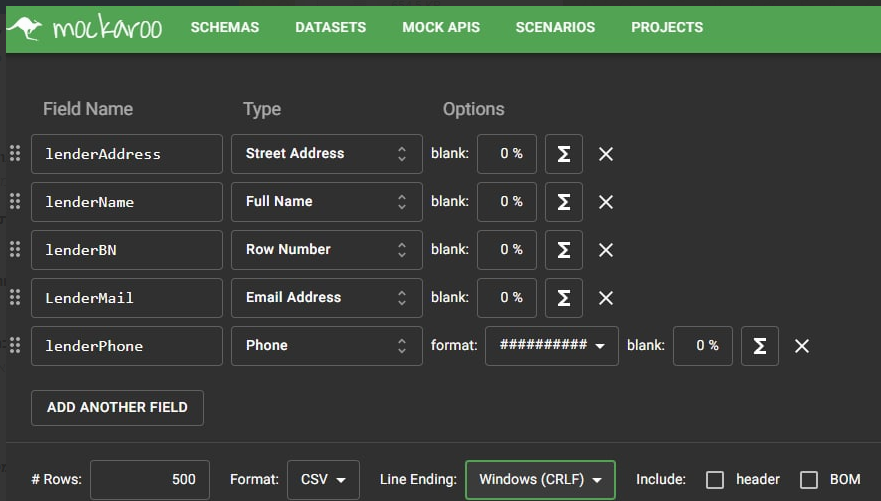
נחולל 100 שורות וניצור קובץ csv שיכיל את הנתונים (בהמשך נטען אותו לpl sql)

תמונה שמכילה טקסט, צילום מסך, מסך, שחור

התיאור נוצר באופן אוטומטי

עבור ישות lender נחולל מידע עבור כל השדות שלו כך שכל שדה יאותחל במידע בהתאם לדרישה (כתובת, שם מלא וכו')

נחולל את הנתונים וניצור קובץ csv שיכיל את הנתונים (בהמשך נטען אותו לpl/sql)



### טעינת קבצי csv ל plsql:

כעת נטען את קבצי csv שיצרנו ב Mockaroo לpl/sql.

כיון שקבצי csv נוצרו עם headers המערכת מזהה את השדות של הישויות (במרכז בצד שמאל).

נטען את הקבצים של lender ושל bank.

לאחר טעינת הקבצים נבצע שאילתה להצגת הנתונים ואכן כל הישויות יוצגו כמצופה.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

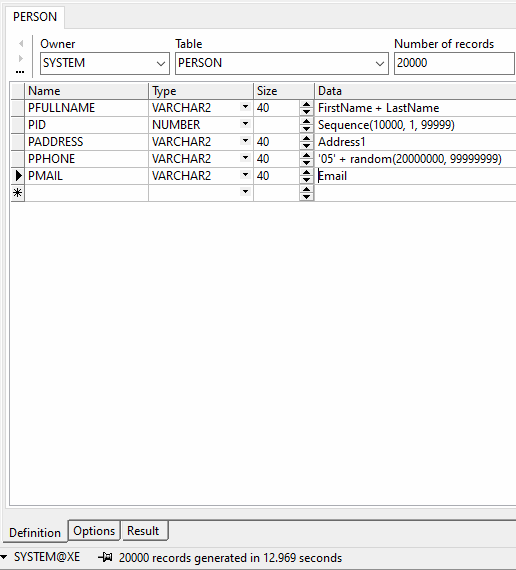
התיאור נוצר באופן אוטומטי

### יצירת ישויות ע"י (plsql) data generator:

נייצר ישויות עבור טבלת Person.

ניצור 20,000 ישויות ונזין בשדות מידע לפי דרישה (שם פרטי + שם משפחה, מספר בטווח מוגדר וכו').

נשתמש בתבניות ופונקציות מוכנות מראש מתוך ה data generator.



כעת נציג את הנתונים מטבלת person.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

נייצר ישויות עבור טבלת President.

ניצור 500 ישויות ונזין בשדות מידע לפי דרישה (במקרה הנ"ל כיון שיש ירושה מ Person נידרש להזין שדה אחד בלבד).

נשתמש בתבניות ופונקציות מוכנות מראש מתוך ה data generator.

כיון שישנה ירושה נשתמש בפונקציה "List" שמאפשרת לנו "לשדך" בין President לישות Person קיימת.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

כעת נציג את הנתונים שיצרנו בעזרת שאילתה.

תמונה שמכילה טקסט

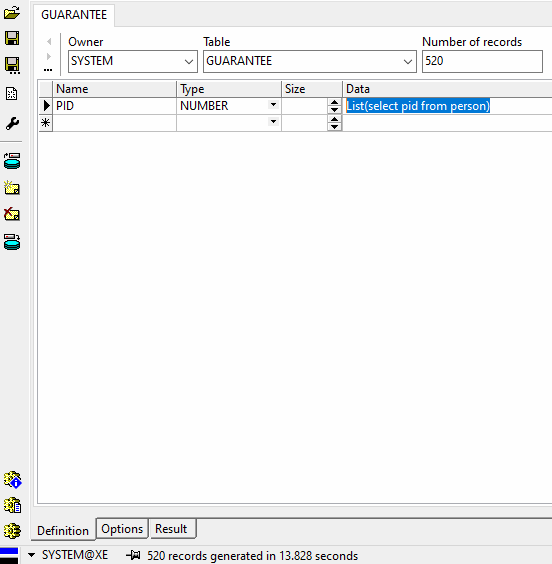
התיאור נוצר באופן אוטומטי

נייצר ישויות עבור טבלת Guarantee.

ניצור 500 ישויות (סדר גודל) ונזין בשדות מידע לפי דרישה (במקרה הנ"ל כיון שיש ירושה מ Person נדרש להזין שדה אחד בלבד).

נשתמש בתבניות ופונקציות מוכנות מראש מתוך ה data generator.

כיון שישנה ירושה נשתמש בפונקציה "List" שמאפשרת לנו "לשדך" בין Guarantee לישות Person קיימת.



כעת בעזרת שאילתה נציג את הנתונים שבטבלת Guarantee.

תמונה שמכילה טקסט

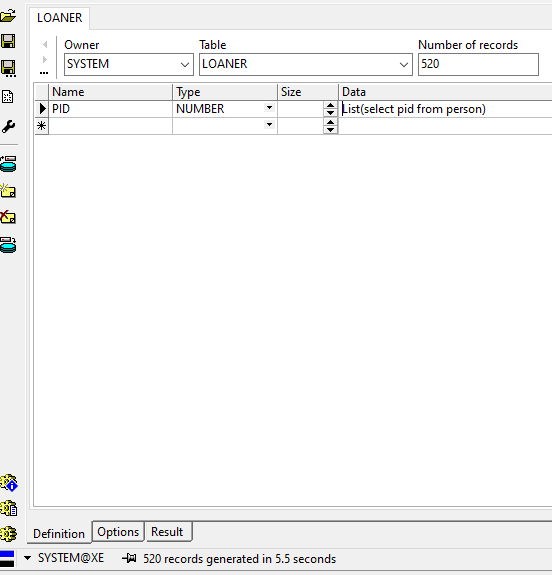
התיאור נוצר באופן אוטומטי

נייצר ישויות עבור טבלת Loaner.

ניצור 500 ישויות (סדר גודל) ונזין בשדות מידע לפי דרישה (במקרה הנ"ל כיון שיש ירושה מ Person נדרש להזין שדה אחד בלבד).

נשתמש בתבניות ופונקציות מוכנות מראש מתוך ה data generator.

כיון שישנה ירושה נשתמש בפונקציה "List" שמאפשרת לנו "לשדך" בין Loaner לישות Person קיימת.



כעת נציג את המידע מטבלת Loaner ע"י שאילתה.

תמונה שמכילה טקסט

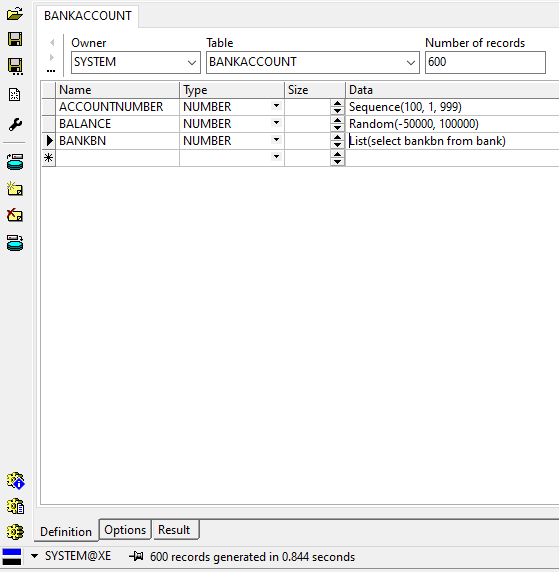
התיאור נוצר באופן אוטומטי

נייצר ישויות עבור טבלת BankAccount.

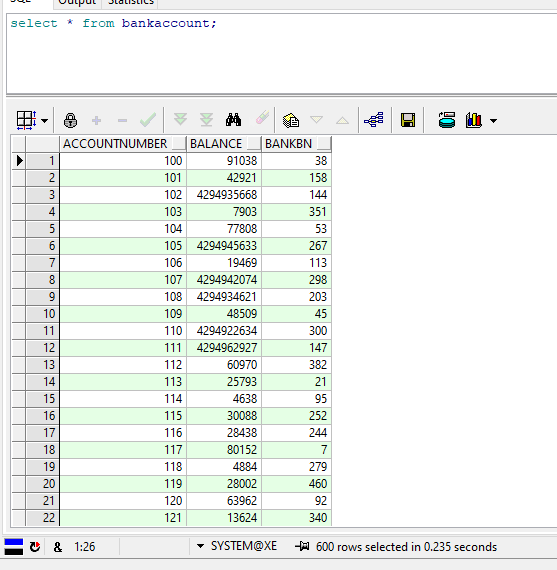
ניצור 500 ישויות (סדר גודל) ונזין בשדות מידע לפי דרישה (מספר בטווח מוגדר וכו' ).

נשתמש בתבניות ופונקציות מוכנות מראש מתוך ה data generator.

כיון ש BankAccount ישות חלשה ל Bank נשתמש בפונקציה "List" שמאפשרת לנו "לשדך" בין BankAccount לישות Bank קיימת.



כעת ע"י שאילתה נציג את נתוני טבלת BankAccount.



נייצר ישויות עבור טבלת Payment.

ניצור 500 ישויות (סדר גודל) ונזין בשדות מידע לפי דרישה (מספר בטווח מוגדר, תאריך הגיוני להתחלה ולסיום וכו' ).

נשתמש בתבניות ופונקציות מוכנות מראש מתוך ה data generator.

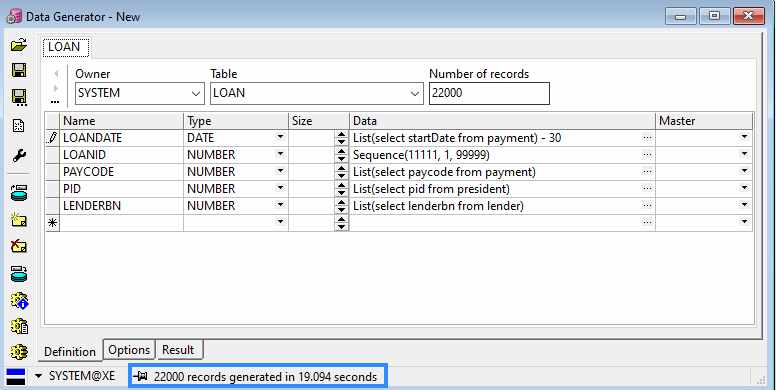
כיון ש Payment מכילה ID של Loaner נשתמש בפונקציה "List" שמאפשרת לנו "לשדך" בין Payment לישות Loaner קיימת.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

בעזרת שאילתה נציג את הנתונים מטבלת Payment.

תמונה שמכילה שולחן

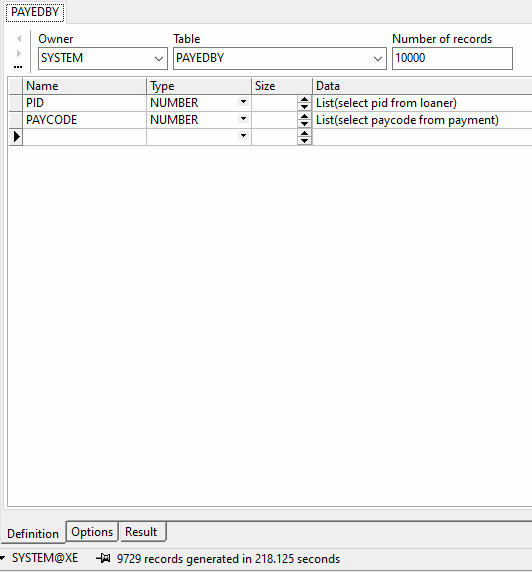
התיאור נוצר באופן אוטומטי

נייצר ישויות עבור טבלת PayedBy.

ניצור 500 ישויות (סדר גודל) ונזין בשדות מידע לפי דרישה.

נשתמש בתבניות ופונקציות מוכנות מראש מתוך ה data generator.

כיון ש PayedBy מכילה ID של Loaner ושל Payment נשתמש בפונקציה "List" שמאפשרת לנו "לשדך" בין Payment לישויות Loaner ו Payment קיימות.



נייצר ישויות עבור טבלת BelongsTo.

ניצור ישויות ונזין בשדות מידע לפי דרישה.

נשתמש בתבניות ופונקציות מוכנות מראש מתוך ה data generator.

כיון ש BelongsTo מכילה ID של Loaner נשתמש בפונקציה "List" שמאפשרת לנו "לשדך" בין Payment לישויות Loaner קיימת.תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

## גיבוי ואחזור הנתונים:

### גיבוי הנתונים:

נגבה את הנתונים דרך pl/sql -> tools -> Export Tables

נבחר את הטבלאות הרצויות לגיבוי (בחרנו את כל הטבלאות שיצרנו)

נבצע Export ל sql script.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

### אחזור המידע:

כעת נבצע דוגמא לשחזור נתונים ע"י שימוש בקובץ הגיבוי:

כך נראה קובץ הגיבוי – SQL למידע שייצרנו.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

נציג את המידע בטבלת Bank:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

נמחק את 100 הרשומות האחרונות מטבלת Bank:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

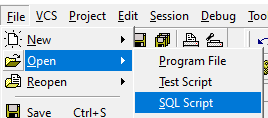
נציג את המידע לאחר המחיקה:

(ניתן לראות שנותרו 400 שורות מתוך 500 שהיו לפני המחיקה)

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

כעת נשתמש בגיבוי לצורך אחזור הרשומות שנמחקו:



נציג את הרשומות מחדש.

ניתן לראות כי כעת ישנן 500 רשומות והמידע אכן אוחזר.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

## **שאילתות SQL**:

1. פרטי המלווים שנתנו הלוואות בסכום כולל העולה על X (בדוגמא הנ"ל - 3000000)

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. הצגת קוד התשלומים בהם ללווה יש בעו"ש סכום גדול יותר מאשר סכום ההלוואה שלקח

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. שמות ות.ז הלווים שלקחו יותר מ 1000 הלוואות

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. ת.ז + ממוצע ההלוואות של כל הנשיאים שחתמו על יותר מ-10 הלוואות וממוצע ההלוואות שעליהן חתמו גבוה מ -70,000

תמונה שמכילה טקסט

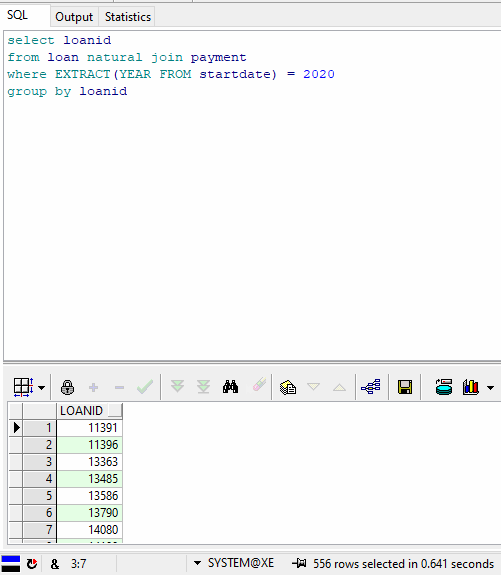
התיאור נוצר באופן אוטומטי

1. קוד (מק"ט) כל ההלוואות שהלווה שלהם מנהל חשבון בבנק מספר 1

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. קוד (מק"ט) כל ההלוואות שנלקחו בשנה מסוימת (בדוגמא הנ"ל - 2020)



1. ת.ז הערבים שסכום ההלוואות עליהן הם חתומים גדול מסכום נתון (בדוגמא הנ"ל 1,000000000(

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. פרטי הערבים שחתומים על יותר מ X הלוואות (בדוגמא הנ"ל - 4)  
     
   תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי

## אינדקסינג:

An index is a performance-tuning method of allowing faster retrieval of records. An index creates an entry for each value that appears in the indexed columns.

### טבלת השוואה לפני – אחרי האינדקסים:

בחנו אינדקסים שונים עבור שאילתות שונות וכן אינדקסים שונים עבור אותן שאילתות.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מס' שאילתה | זמן ללא אינדקס | זמן עם אינדקס | אינדקס חיובי / שלילי | אחוז השינוי |
| 1 | 0.172 | 0.204 | שלילי | +18% |
| 4 | 0.172 | 0.453 | שלילי | +160% |
| 4 | 0.172 | 0.157 | חיובי | -9% |
| 5 | 28.281 | 6.702 | חיובי | -76% |
| 6 | 0.641 | 0.234 | חיובי | -63% |
| 7 | 9.579 | 1.859 | חיובי | -80% |

### אינדקסים יעילים:

עבור שאילתה (7):

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

עבור שאילתה (4)

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

עבור שאילתה (5)

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

עבור שאילתה (6)

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

### אינדקסים לא-יעילים:

עבור שאילתה (1):

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

עבור שאילתה (4):

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

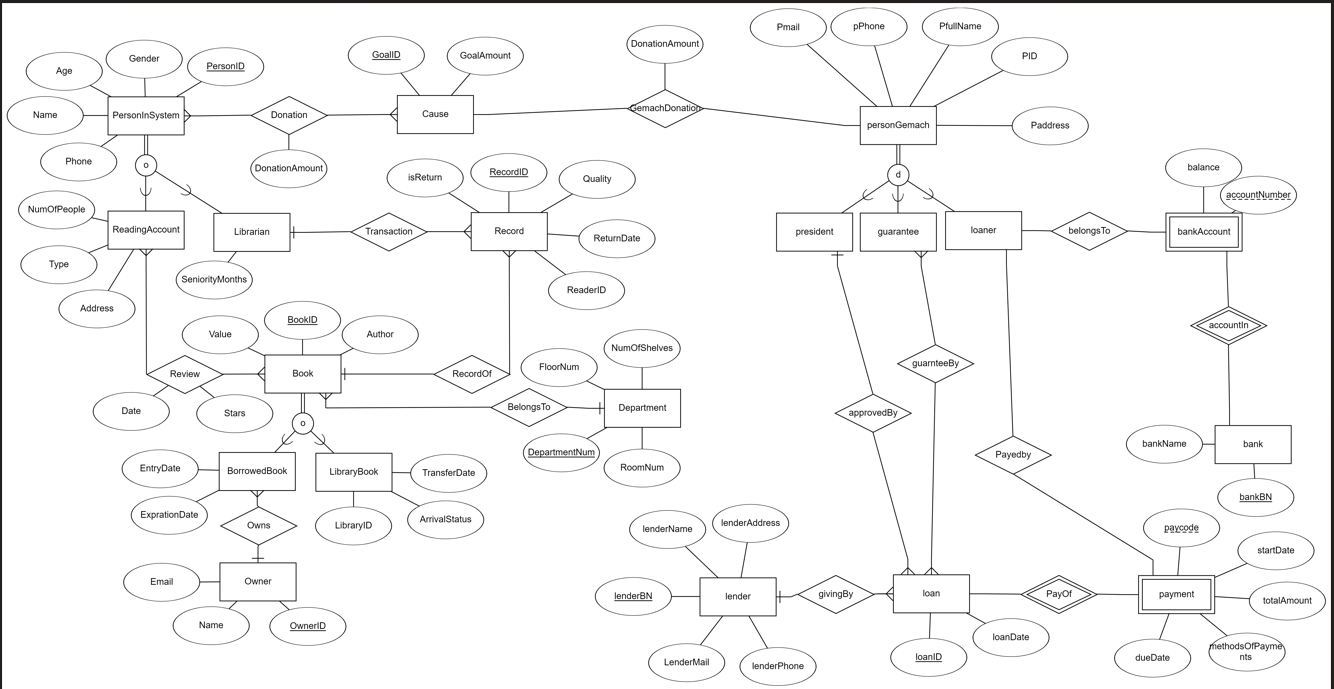
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

# שלב ג

## אינטגרציה:

אינטגרציה של בסיס הנתונים שלנו עם בסיס הנתונים של גמ"ח כספים של הזוג אליהו רוזנפלד ונבו כהן:



# שלב ד

## Views:

### View 1:

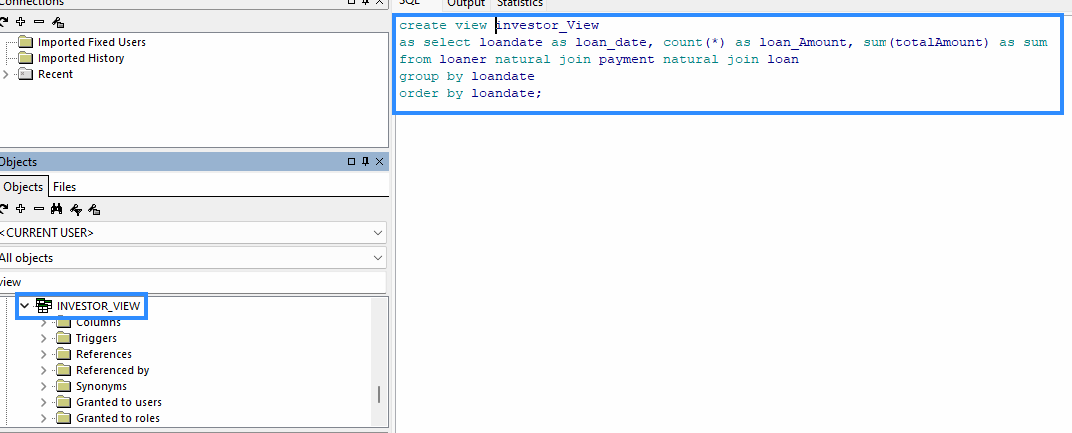
את view זה יצרנו עבור אדם שרוצה "להשקיע" בגמ"ח (תרומת כספים).

ב view זה מוצגות ההלוואות לפי ימים כך שעבור כל יום יוצגו השדות הבאים:

תאריך,

מספר ההלוואות שנלקחו באותו יום,

הסכום הכולל של סכום ההלוואות שנלקחו באותו יום (באגורות).



#### שאילתה מס' 1 עבור view 1:

ניתן לראות שכאשר אנו מציגים נתונים לפי סינון מה view שנשמר (סכום ההלוואות לאותו יום גדול מ2,000,000 אגורות) נקבל את כל השדות שהוזכרו לעיל.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

#### שאילתה מס' 2 עבור view 1:

ניתן לראות שכאשר אנו מציגים נתונים לפי סינון מה view שנשמר (הלוואות שנלקחו החל משנת 2000) נקבל את כל השדות שהוזכרו לעיל.

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

### View 2:

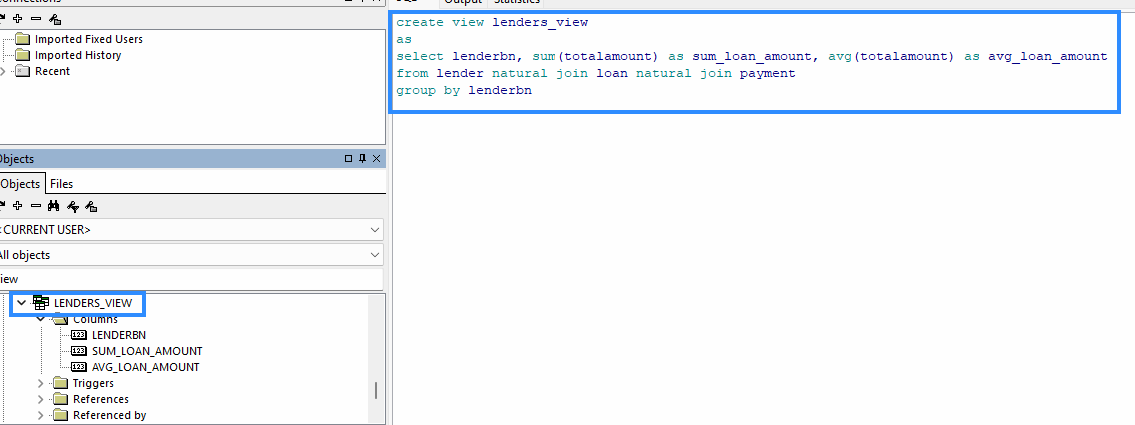
את view זה יצרנו בכדי לבצע מעקב וניתוח פעולות המלווה (גמ"ח או לחליפין נציג הגמ"ח).

ב view זה מוצגים נתונים עבור פעולות של המלווה כך שעבור כל מלווה יוצגו השדות הבאים:

מספר מזהה של המלווה,

הסכום הכולל של אותן ההלוואות שנתן המלווה או לחליפין שעליהן חתם (באגורות).

הממוצע הכולל של סכום אותן ההלוואות שנתן המלווה או לחליפין שעליהן חתם (באגורות).



#### שאילתה מס' 1 עבור view 2:

בשאילתה זו נציג את השדות – מספר המלווה, סכום ההלוואות שנתן / אישר, ובתנאי שהסכום גדול מ 4,000 (₪)

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

#### שאילתה מס' 2 עבור view 2:

בשאילתה זו נציג את – מספר מזהה של הלווה, ממוצע סכום ההלוואות אותם נתן / אישר, ובתנאי שהממוצע גדול מ 75000 (אגורות).

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

## Procedures

### Procedure 1:

ב procedure זה אנחנו מקבלים כפרמטר חודש מסוים (1-12) ומחזירים נתונים על ההלוואות שהתבצעו לאורך השנים בחודש זה.

את Procedure זו יצרנו בכדי לאפשר ניתור של פעילות הגמ"ח לפי חודשים וכך למשל ניתן יהיה לצפות מראש חודשים "עמוסים" יותר ולהיערך אליהם בהתאם.

תמונה שמכילה טקסט

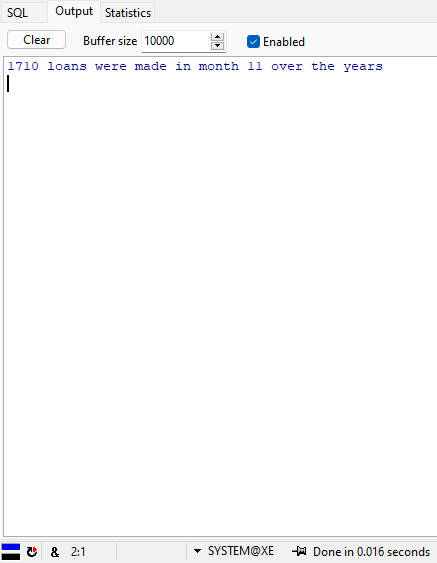
התיאור נוצר באופן אוטומטי

קריאה ל procedure שיצרנו:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

ונקבל כפלט:



### Procedure 2:

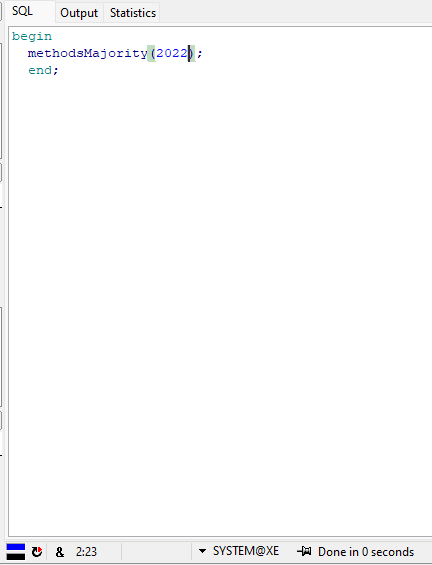
ב Procedure זו אנו מקבלים כפרמטר שנה (1970 – today) ומחזירים כפלט האם באותה שנה התבצעו יותר הלוואות שהתקבלו במזומן (להלן cash) או בצ'ק (להלן check).

Procedure זה מיועד עבור בקרה של תזרים המזומנים של הגמ"ח, כך שאם גמ"ח מסוים הגדיר כיעד שרוב התזרים יהיה בצ'קים, ע"י שימוש ב Procedure זו הוא יוכל לבדוק זאת בקלות.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

קריאה ל procedure שיצרנו:



פלט ה Procedure:



## Functions:

### Functions 1:

Function זו מקבלת כפרמטר סכום כסף ומחזירה את אחוז ההלוואות שמעל (או שווה) סכום זה ביחס לכלל ההלוואות שבוצעו בגמ"ח.

Function זו מאפשר לנו לזהות נתונים סטטיסטיים עבור הגמ"ח, כך למשל ניתן לזהות חציון בסכום ההלוואות.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הרצת ה Function עם פרמטר (באגורות):

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

קבלת הפלט:



### Function 2:

Function זו מקבלת כפרמטרים טווח תאריכים בשנים ומחזירה את מספר ההלוואות שבוצעו בטווח זה.

Function זו מיועדת עבור ניתוח פעולות בגמ"ח, כך למשל ניתן לזהות מגמת עליה בטווח שנים מסוים.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הרצת ה Function:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

קבלת הפלט:



## Triggers:

### Triggers 1:

Trigger זה נועד "לעטוף" פעולות בסיסיות ב Log מתאים ובך להפוך את השימוש בפעולות לנוח יותר (כפי שניתן לראות בהמשך בדוגמת ההרצה)

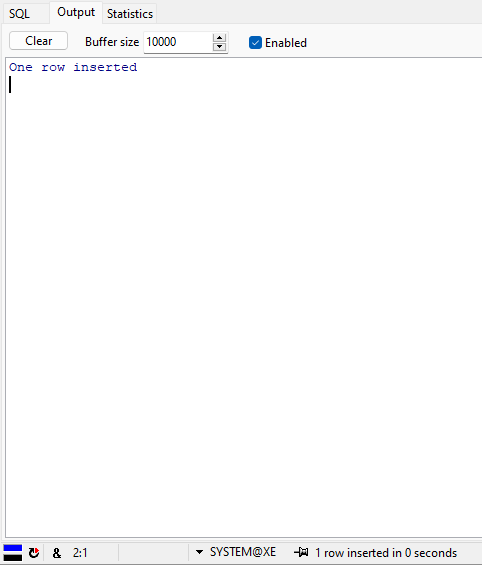
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הפעלה לאחר יצירת ה Trigger:



פלט המערכת:



### Trigger 2:

Trigger זה נוצר בכדי לבצע פיקוח על שדה ה ID בטבלת Person כך שהוא יוצר באופן עקבי לפי סדר קבוע מראש.

ראשית ניצור משתנה עזר בתור רצף שלפיו יקבע שדה ה ID:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

כעת ניצור את ה Trigger עצמו (שמשתמש במשתנה שיצרו קודם):

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

בדיקת הטריגר ע"י הכנסת ערך "שגוי" בשדה ID:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

כעת נציג את הרשומה שנוספה, ניתן לראות שהיא אכן נוספה אך עם שדה ID אחר מזה שהופיע ב Values בפעולת ה Insert (הערך בפועל נקבע לפי משתנה הרץ שהגדרנו עבור ה Trigger):

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

## Git:

<https://github.com/Tzur-Eitan/DB_PROJECT>

Microsoft Windows [Version 10.0.22000.675]

(c) Microsoft Corporation. All rights reserved.

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj>git clone https://github.com/Tzur-Eitan/DB\_PROJECT.git

Cloning into 'DB\_PROJECT'...

remote: Enumerating objects: 11, done.

remote: Counting objects: 100% (11/11), done.

remote: Compressing objects: 100% (9/9), done.

remote: Total 11 (delta 1), reused 11 (delta 1), pack-reused 0

Receiving objects: 100% (11/11), 1.13 MiB | 285.00 KiB/s, done.

Resolving deltas: 100% (1/1), done.

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj>git status

On branch main

nothing to commit, working tree clean

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj>cd DB\_PROJECT

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git add .

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git commit -m "stage 2"

[main c38c4a2] stage 2

10 files changed, 322486 insertions(+)

create mode 100644 "\327\251\327\234\327\221 \327\221/MOCK\_DATA.csv"

create mode 100644 "\327\251\327\234\327\221 \327\221/bank2.csv"

create mode 100644 "\327\251\327\234\327\221 \327\221/lender2.csv"

create mode 100644 "\327\251\327\234\327\221 \327\221/payedby\327\222\327\231\327\221\327\225\327\231.sql"

create mode 100644 "\327\251\327\234\327\221 \327\221/\327\222\327\231\327\221\327\225\327\231.sql"

create mode 100644 "\327\251\327\234\327\221 \327\221/\327\222\327\231\327\221\327\225\327\231.~sql"

create mode 100644 "\327\251\327\234\327\221 \327\221/\327\222\327\231\327\221\327\225\327\231\_\327\251\327\240\327\231.sql"

create mode 100644 "\327\251\327\234\327\221 \327\221/\327\223\327\225\327\222\327\236\327\220 \327\234\327\222\327\231\327\221\327\225\327\231/\327\223\327\225\327\222\327\236\327\220 \327\224\327\251\327\247\327\225\327\224.sql"

create mode 100644 "\327\251\327\234\327\221 \327\221/\327\251\327\234\327\221 \327\221.docx"

create mode 100644 "\327\251\327\234\327\221 \327\221/\327\251\327\234\327\221 \327\221.pdf"

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git status

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git push

Enumerating objects: 14, done.

Counting objects: 100% (14/14), done.

Delta compression using up to 8 threads

Compressing objects: 100% (13/13), done.

Writing objects: 100% (13/13), 6.33 MiB | 1.94 MiB/s, done.

Total 13 (delta 1), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (1/1), done.

To https://github.com/Tzur-Eitan/DB\_PROJECT.git

fbe4803..c38c4a2 main -> main

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git status

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git add .

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git commit -m "stage 3"

[main b2c1e5d] stage 3

11 files changed, 103 insertions(+)

create mode 100644 "\327\251\327\234\327\221 \327\222/DSD Copy.erdplus"

create mode 100644 "\327\251\327\234\327\221 \327\222/Elyasaf.txt"

create mode 100644 "\327\251\327\234\327\221 \327\222/Nevo.erdplus.erdplus"

create mode 100644 "\327\251\327\234\327\221 \327\222/United.erdplus.erdplus.erdplus"

create mode 100644 "\327\251\327\234\327\221 \327\222/WhatsApp Image 2022-05-17 at 13.47.53.jpeg"

create mode 100644 "\327\251\327\234\327\221 \327\222/image (1).png"

create mode 100644 "\327\251\327\234\327\221 \327\222/image.png"

create mode 100644 "\327\251\327\234\327\221 \327\222/integrated.png"

create mode 100644 "\327\251\327\234\327\221 \327\222/\327\220\327\231\327\240\327\230\327\222\327\250\327\246\327\231\327\224.docx"

create mode 100644 "\327\251\327\234\327\221 \327\222/\327\251\327\234\327\221 \327\222.docx"

create mode 100644 "\327\251\327\234\327\221 \327\222/\327\251\327\234\327\221 \327\222.pdf"

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git status

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git push

Enumerating objects: 15, done.

Counting objects: 100% (15/15), done.

Delta compression using up to 8 threads

Compressing objects: 100% (14/14), done.

Writing objects: 100% (14/14), 7.69 MiB | 1.45 MiB/s, done.

Total 14 (delta 1), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (1/1), done.

To https://github.com/Tzur-Eitan/DB\_PROJECT.git

c38c4a2..b2c1e5d main -> main

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>git status

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\אליסף\Desktop\אליסף\אא לימודים\שנה ג\סימסטר ב\מיניפ\sqlProj\DB\_PROJECT>