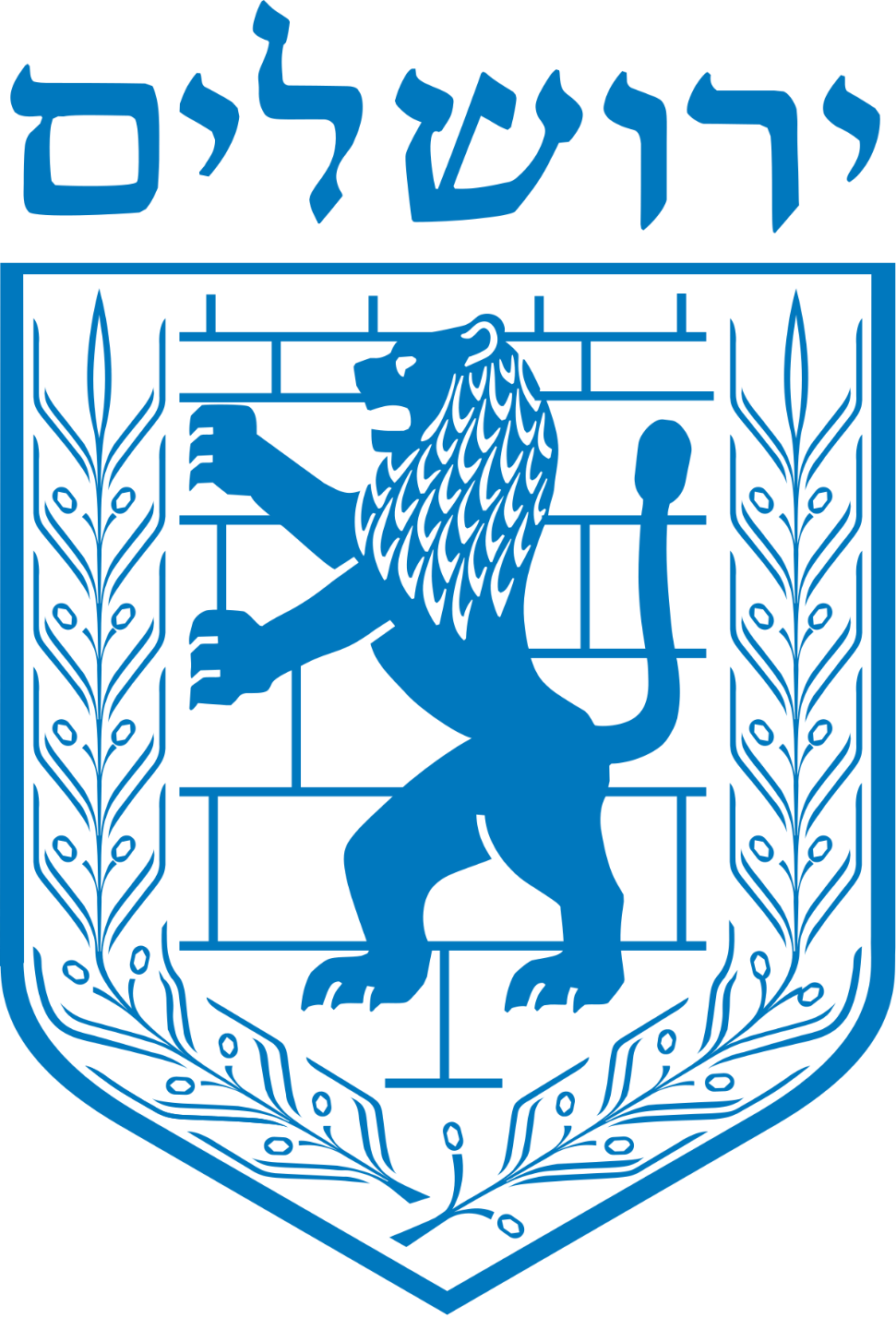
|  |
| --- |
|  |

מיני פרויקט בבסיסי נתונים – עירייה



**נועם שוובר – 214120032**

**נריה ניימן –214186801**

# תוכן עניינים

[תוכן עניינים 2](#_Toc103703059)

[חלק א' 3](#_Toc103703060)

[דיאגרמת ER 3](#_Toc103703061)

[דיאגרמת DSD (יחסים) 3](#_Toc103703062)

[הסבר הישויות 4](#_Toc103703063)

[חלק ב' 13](#_Toc103703064)

[אכלוס הטבלאות 13](#_Toc103703065)

[שאילתות מיוחדות 33](#_Toc103703066)

[שאילתא מס' 1 33](#_Toc103703067)

[שאילתא מס' 2 34](#_Toc103703068)

[שאילתא מס' 3 35](#_Toc103703069)

[שאילתא מס' 4 36](#_Toc103703070)

[שאילתא מס' 5 37](#_Toc103703071)

[שאילתא מס' 6 38](#_Toc103703072)

[שאילתא מס' 7 39](#_Toc103703073)

[שאילתא מס' 8 40](#_Toc103703074)

[אינדקסים 41](#_Toc103703075)

[חלק ג' 44](#_Toc103703076)

[דיאגרמות 44](#_Toc103703077)

[התאמות 46](#_Toc103703078)

[אילוצים – Constraints 52](#_Toc103703079)

[שאילתות 53](#_Toc103703080)

[חלק ד' 55](#_Toc103703081)

[תצפיות – Views 55](#_Toc103703082)

[פונקציות – Functions 57](#_Toc103703083)

[פרוצדורות – Procedures 59](#_Toc103703084)

[טריגרים – Triggers 61](#_Toc103703085)

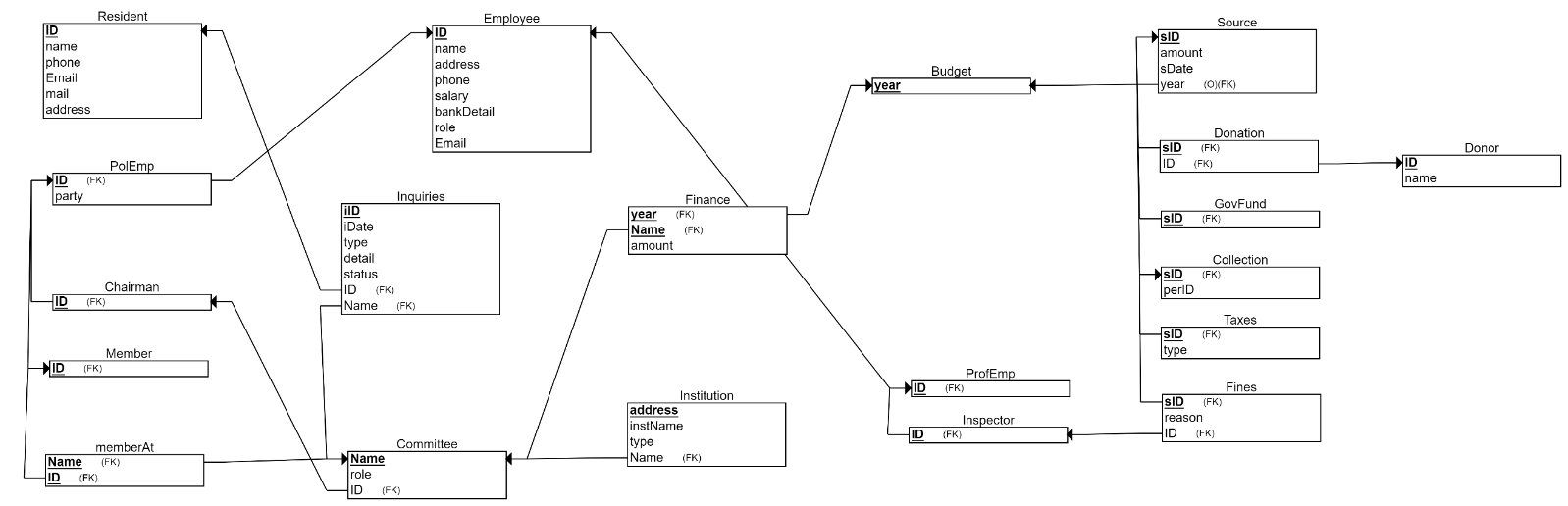
[חלק ה' 63](#_Toc103703086)

[גרפים 63](#_Toc103703087)

[דו"חות 64](#_Toc103703088)

**חלק א'**

**דיאגרמת ER**

**דיאגרמת DSD (יחסים)**

תמונה שמכילה טקסט, מפה, מקורה

התיאור נוצר באופן אוטומטי

## הסבר הישויות

1. **Employee**

מייצג עובד כללי – בין אם גר בעירייה, בין אם לא.

תכונות:

* ID – ת"ז (מפתח)
* Name – שם
* Address – כתובת
* Phone – מספר פלאפון
* Salary – משכורת
* Bank Details – פרטי בנק
* Role – תפקיד
* Email – כתובת מייל

Create Table:

CREATE TABLE Employee

(

ID INT NOT NULL,

name VARCHAR(30) NOT NULL,

address VARCHAR(30) NOT NULL,

phone VARCHAR(11) NOT NULL,

salary INT NOT NULL,

bankDetail VARCHAR(20) NOT NULL,

role VARCHAR(20) NOT NULL,

Email VARCHAR(30) NOT NULL,

PRIMARY KEY (ID)

);

1. **ProfEmp**

מייצג עובד מקצועי – יכול לגור בעירייה ויכול שלא. יורש מעובד כללי.

תכונות:

* ID – ת"ז (מפתח, מפתח זר של Employee)

CREATE TABLE profEmp

(

ID INT NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (ID) REFERENCES Employee(ID)

);

1. **Resident**

מייצג תושב בעירייה.  
תכונות:

* ID – ת"ז (מפתח)
* Name – שם
* Phone – מספר פלאפון
* Email – כתובת מייל
* Mail – כתובת דואר
* Address – כתובת

Create Table:

CREATE TABLE Resident

(

ID INT NOT NULL,

name VARCHAR(30) NOT NULL,

phone VARCHAR(11) NOT NULL,

Email VARCHAR(30) NOT NULL,

mail VARCHAR(30) NOT NULL,

address VARCHAR(30) NOT NULL,

PRIMARY KEY (ID)

);

1. **PolEmp**

מייצג עובד פוליטי – בהכרח גר בעירייה. יורש מעובד כללי.  
תכונות:

* ID – ת"ז (מפתח, מפתח זר של Employee ושל Resident)
* Party – מפלגה

Create Table:

CREATE TABLE polEmp

(

party VARCHAR(20) NOT NULL,

ID INT NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (ID) REFERENCES Employee(ID),

FOREIGN KEY (ID) REFERENCES Resident(ID)

);

1. **Budget**

מייצג תקציב שנתי.  
תכונות:

* year – שנת התקציב (מפתח)

Create Table:

CREATE TABLE Budget

(

year INT NOT NULL,

PRIMARY KEY (year)

);

1. **Source**

מייצג מקור כלכלי לתקציב.   
תכונות:

* sID – מספר סידורי של התקציב (מפתח)
* sDate – תאריך קבלת המקור.
* Amount – כמות המקור.
* Year – שנת התקציב (מפתח זר של Budget)

Create Table:

CREATE TABLE Source

(

amount INT NOT NULL,

sDate DATE NOT NULL,

sID INT NOT NULL,

year INT,

PRIMARY KEY (sID),

FOREIGN KEY (year) REFERENCES Budget(year)

);

1. **Collection**

מייצג אסיפת כסף כלשהי מאדם (לאו דווקא מתושב העירייה). יורש מSource.  
תכונות:

* sID – מספר סידורי של התקציב (מפתח, מפתח זר של Source)
* ID – מייצג את תעודת הזהות של האדם ממנו נאסף הכסף. (לא מקושר כיוון שיכול להיות מחוץ לעירייה)

Create Table:

CREATE TABLE Collection

(

ID INT NOT NULL,

sID INT NOT NULL,

PRIMARY KEY (sID),

FOREIGN KEY (sID) REFERENCES Source(sID)

);

1. **GovFund**

מייצג תקצוב מהממשלה. יורש מSource.  
תכונות:

* sID – מספר סידורי של התקציב (מפתח, מפתח זר של Source)

Create Table:

CREATE TABLE GovFund

(

sID INT NOT NULL,

PRIMARY KEY (sID),

FOREIGN KEY (sID) REFERENCES Source(sID)

);

1. **Taxes**

מייצג מיסים שנאספו. יורש מCollection.  
תכונות:

* sID – מספר סידורי של התקציב (מפתח, מפתח זר של Collection)
* Type – סוג מסים.

Create Table:

CREATE TABLE Taxes

(

type VARCHAR(10) NOT NULL,

sID INT NOT NULL,

PRIMARY KEY (sID),

FOREIGN KEY (sID) REFERENCES Collection(sID)

);

1. **Donor**

מייצג תורם לעירייה.  
תכונות:

* ID – ת"ז (מפתח)
* Name – שם.

Create Table:

CREATE TABLE Donor

(

name VARCHAR(30) NOT NULL,

ID INT NOT NULL,

PRIMARY KEY (ID)

);

1. **Donation**

מייצג תרומה. יורש מSource.  
תכונות:

* sID – מספר סידורי של התקציב (מפתח, מפתח זר של Source)
* dID – ת"ז של תורם (מפתח זר של Donor)

Create Table:

CREATE TABLE Donation

(

sID INT NOT NULL,

dID INT NOT NULL,

PRIMARY KEY (sID),

FOREIGN KEY (sID) REFERENCES Source(sID),

FOREIGN KEY (dID) REFERENCES Donor(ID)

);

1. **Inspector**

מייצג פקח בעירייה. יורש מעובד מקצועי.  
תכונות:

* ID – ת"ז (מפתח, מפתח זר של Employee)

Create Table:

CREATE TABLE Inspector

(

ID INT NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (ID) REFERENCES profEmp(ID)

);

1. **Fines**

מייצג קנס שנאסף (בעקבות עבירה). יורש מCollection.  
תכונות:

* sID – מספר סידורי של התקציב (מפתח, מפתח זר של Source)
* Reason – סיבה לקבלת הקנס.
* iID – ת"ז של הפקח (מקושר לInspector)

Create Table:

CREATE TABLE Fines

(

reason VARCHAR(40) NOT NULL,

sID INT NOT NULL,

iID INT NOT NULL,

PRIMARY KEY (sID),

FOREIGN KEY (sID) REFERENCES Collection(sID),

FOREIGN KEY (iID) REFERENCES Inspector(ID)

);

1. **Member**

מייצג חבר וועדה – יורש מעובד פוליטי.  
תכונות:

* ID – ת"ז (מפתח, מפתח זר של PolEmployee)

Create Table:

CREATE TABLE Member

(

ID INT NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (ID) REFERENCES polEmp(ID)

);

1. **Chairman**

מייצג יו"ר ועדה. יורש מעובד פוליטי.  
תכונות:

* ID – ת"ז (מפתח, מפתח זר של Employee ושל Resident)

Create Table:

CREATE TABLE Chairman

(

ID INT NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (ID) REFERENCES polEmp(ID)

);

1. **Committee**

מייצג ועדה של נבחרי הציבור בעירייה. (לדוג' ועדת חינוך, ועדת תרבות)  
תכונות:

* Name – שם הועדה (מפתח)
* Role – תפקיד הוועדה [בפירוט]
* ID – ת"ז של יו"ר הוועדה (מפתח זר של Chairman)

Create Table:

CREATE TABLE Committee

(

Name VARCHAR(30) NOT NULL,

role VARCHAR(30) NOT NULL,

ID INT NOT NULL,

PRIMARY KEY (Name),

FOREIGN KEY (ID) REFERENCES Chairman(ID)

)

1. **Institution**

מייצג מוסד בעירייה – לכל מוסד יש וועדה כלשהי שדואגת לו.   
תכונות:

* Address – כתובת (מפתח)
* instName – שם המוסד
* Type – סוג המוסד.
* commitName – שם הוועדה המטפלת. (מפתח זר של Committee)

Create Table:

CREATE TABLE Institution

(

address VARCHAR(30) NOT NULL,

instName VARCHAR(30) NOT NULL,

type CHAR(10) NOT NULL,

commitName VARCHAR(30) NOT NULL,

PRIMARY KEY (address),

FOREIGN KEY (commitName) REFERENCES Committee(Name)

);

1. **MemberAt**

מייצג קשר בין חברי וועדה לוועדה.   
תכונות:

* Name – שם הוועדה (חלק ממפתח, מפתח זר של Committee)
* ID – ת"ז של חבר הוועדה (חלק ממפתח, מפתח זר של Member)

Create Table:

CREATE TABLE memberAt

(

Name VARCHAR(30) NOT NULL,

ID INT NOT NULL,

PRIMARY KEY (Name, ID),

FOREIGN KEY (Name) REFERENCES Committee(Name),

FOREIGN KEY (ID) REFERENCES Member(ID)

);

1. **Inquiries**

מייצג פנייה מתושב העירייה – כל פנייה מטופלת על ידי ועדה כלשהי.   
תכונות:

* iID – המספר הסידורי של הפנייה (מפתח)
* iDate – תאריך פתיחת הפנייה.
* Type – סוג הפנייה.
* Details – פרטי הפנייה.
* Status – סטטוס הפנייה.
* ID – ת"ז התושב הפונה (מפתח זר של Resident)
* commitName – שם הוועדה (מפתח זר של Commitee)

Create Table:

CREATE TABLE Inquiries

(

iID INT NOT NULL,

iDate DATE NOT NULL,

type VARCHAR(10) NOT NULL,

details VARCHAR(50) NOT NULL,

status CHAR(1) NOT NULL,

ID INT NOT NULL,

commitName VARCHAR(30) NOT NULL,

PRIMARY KEY (iID),

FOREIGN KEY (ID) REFERENCES Resident(ID),

FOREIGN KEY (commitName) REFERENCES Committee(Name)

);

1. **Finance**

מייצג תקצוב שנתי לוועדה, לפי וועדה ושנה.  
תכונות:

* year – שנת התקציב (חלק ממפתח, מפתח זר של Budget).
* Name – שם הוועדה (חלק ממפתח, מפתח זר של Committee).
* Amount – כמות התקציב מהשנה לוועדה.

Create Table:

CREATE TABLE Finance

(

amount INT NOT NULL,

year INT NOT NULL,

Name VARCHAR(30) NOT NULL,

PRIMARY KEY (year, Name),

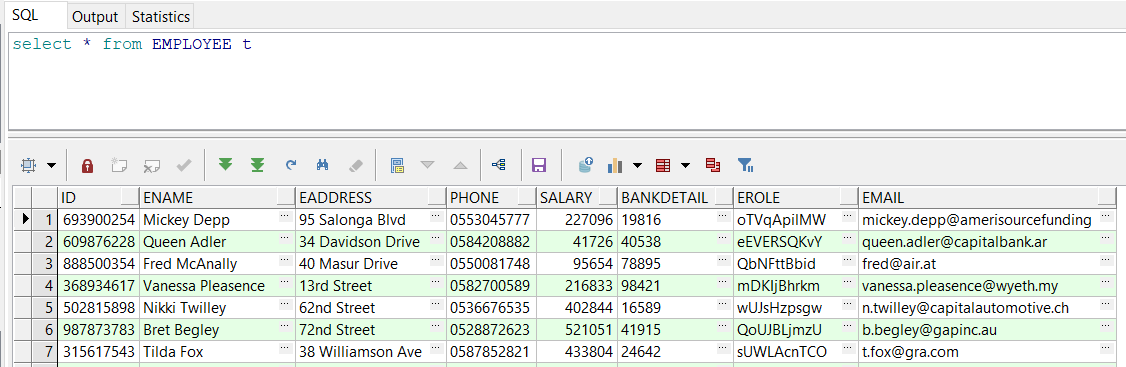
FOREIGN KEY (year) REFERENCES Budget(year),

FOREIGN KEY (Name) REFERENCES Committee(Name)

);

# חלק ב'

## אכלוס הטבלאות

1. **תמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיתמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטי****Employee** – הכנסה באמצעות Data Generator
2. תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי**תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי**תמונה שמכילה טקסט

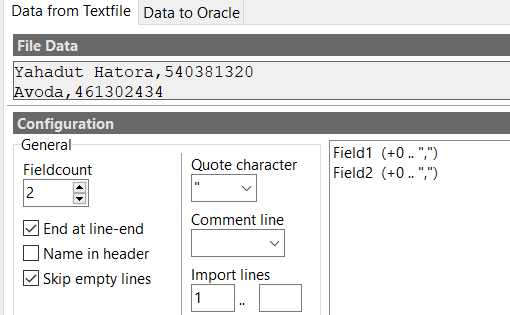
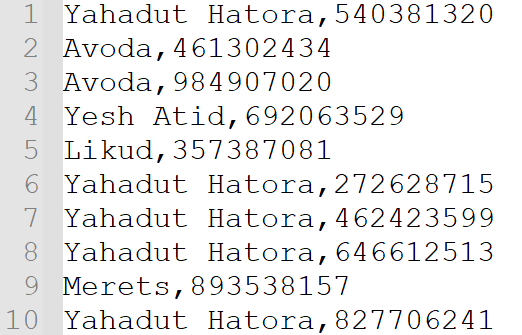
   התיאור נוצר באופן אוטומטי**ProfEmp** – הכנסה באמצעות Text Importer
3. **תמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיResident** – הכנסה באמצעות Data Generator

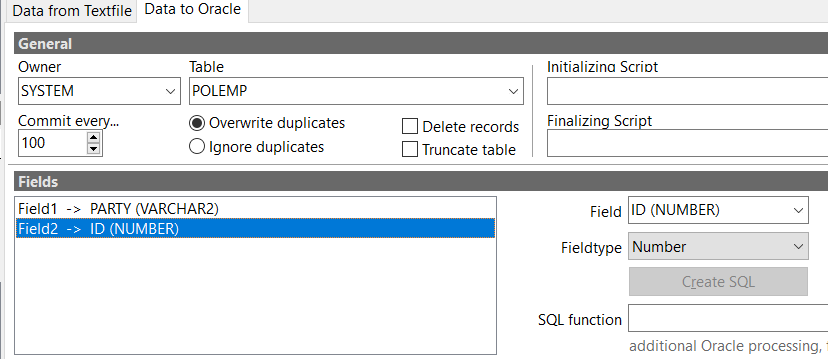
**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

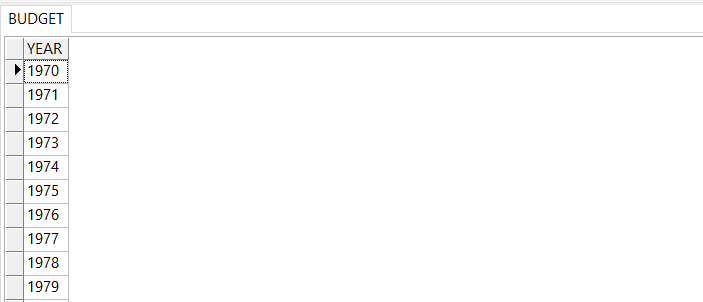
התיאור נוצר באופן אוטומטי=**

1. **PolEmp** – הכנסה באמצעות Text Importer

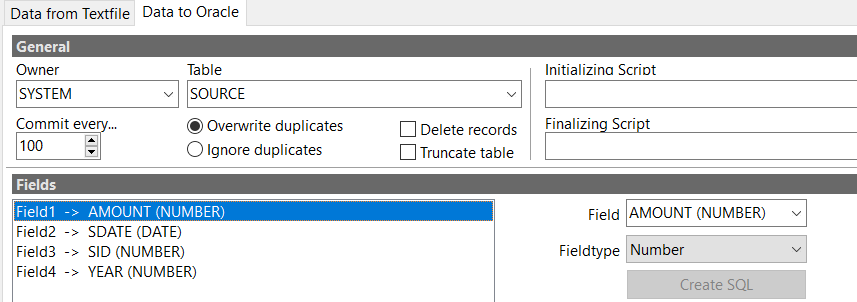
**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

1. **תמונה שמכילה טקסט

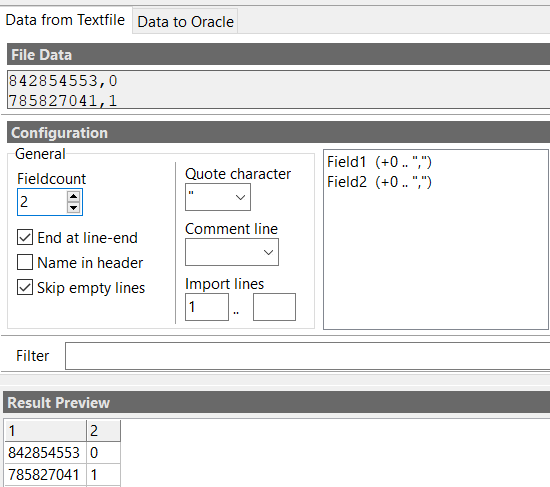
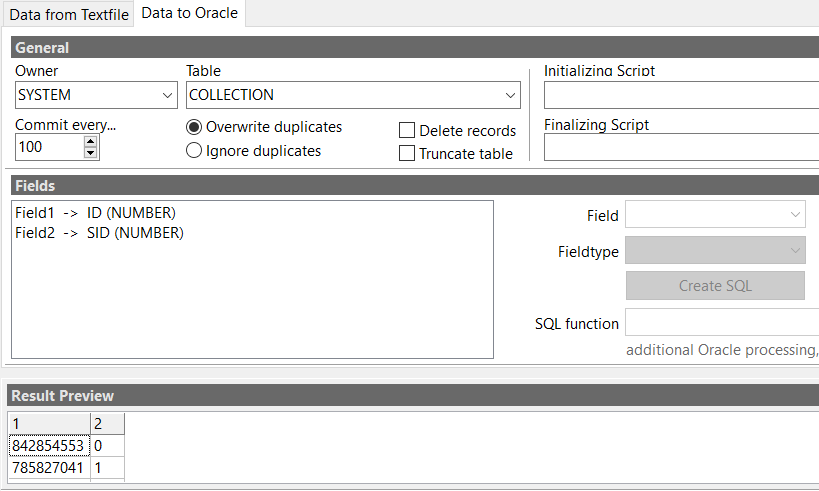
   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

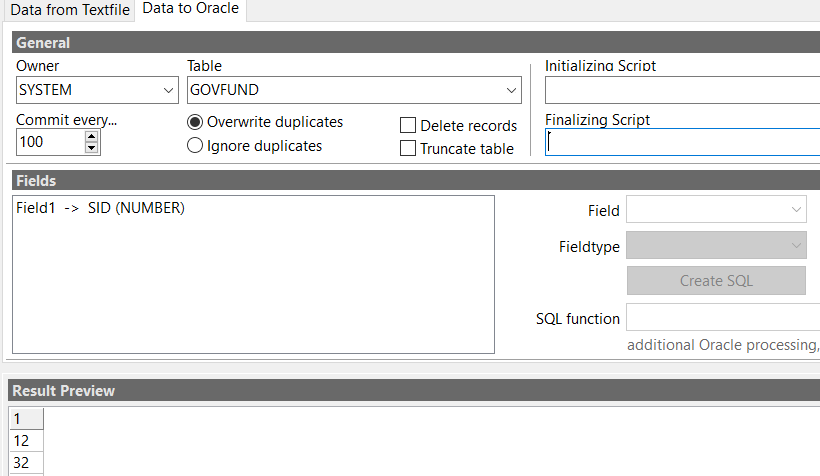
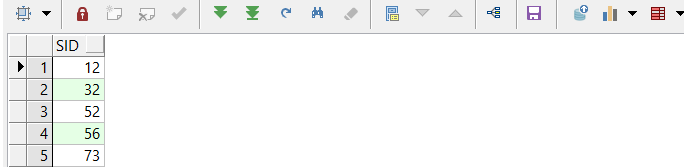
   התיאור נוצר באופן אוטומטיBudget** – הכנסה באמצעות Data Generator
2. **תמונה שמכילה שולחן

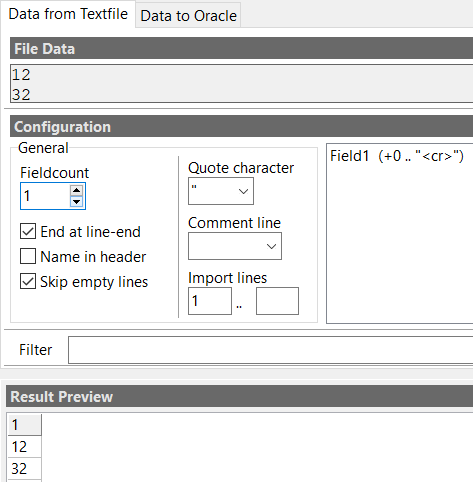
   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

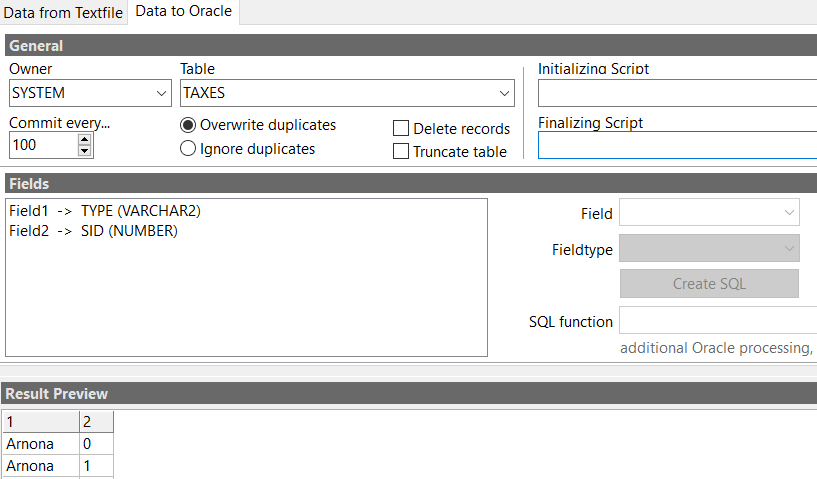
   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיSource** – הכנסה באמצעות Text Importer
3. **תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיCollection** –הכנסה באמצעות Text Importer
4. ****תמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטי**GovFund** – הכנסה באמצעות Text Importer
5. **תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי**תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי**Taxes** – הכנסה באמצעות Text Importer
6. **תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיתמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיDonor** – הכנסה באמצעות Data Generator
7. **תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי**תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי**Donation** – הכנסה באמצעות Text Importer
8. **תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיInspector** – הכנסה באמצעות Data Generator

1. **תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי**תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי**Fines** – הכנסה באמצעות Text Importer
2. **תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיMember** – הכנסה באמצעות Insert
3. **תמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיChairman** – הכנסה באמצעות Insert

1. **תמונה שמכילה טקסט, שולחן

   התיאור נוצר באופן אוטומטיCommittee** – הכנסה באמצעות Insert

**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

1. **תמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיתמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיInstitution** – הכנסה באמצעות Data Generator

1. **תמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיMemberAt** – הכנסה באמצעות Insert

1. **תמונה שמכילה טקסט

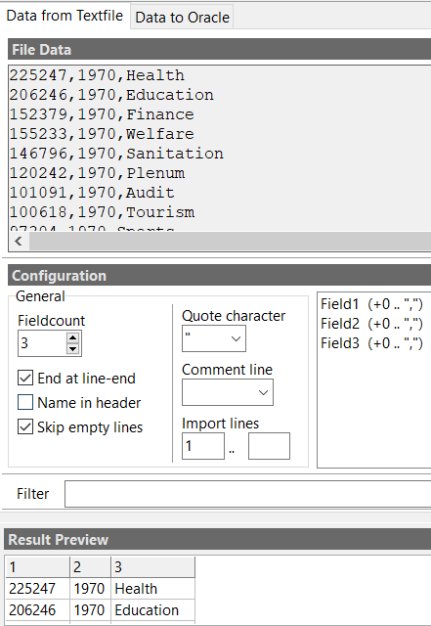
   התיאור נוצר באופן אוטומטיתמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיInquiries** – הכנסה באמצעות Data Generator
2. תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי**תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטי****Finance** – הכנסה באמצעות Text Importer

## שאילתות מיוחדות

שאילתא מס' 1

**תיאור**: אם נקצץ 20% ממשכורות סוגי השונים העובדים או 1% מתקציבי הוועדות, אילו ועדות וסוגי עובדים שלהם נקצץ בתקציב ייתן לנו יותר?  
**קוד**:

SELECT \*

FROM ((SELECT EROLE AS NAME, SUM(SALARY) - SUM(SALARY \* 0.8) AS SAVING

FROM EMPLOYEE

GROUP BY EROLE)

UNION

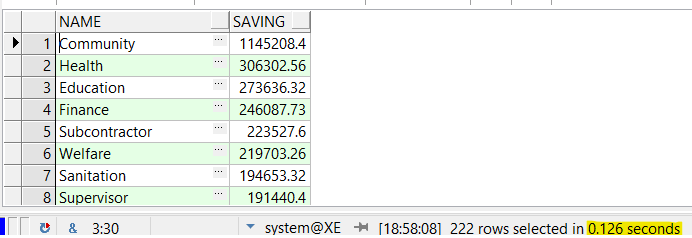
(SELECT C.CNAME AS NAME, SUM(F.AMOUNT) - SUM(AMOUNT \* 0.99) AS SAVING

FROM COMMITTEE C, FINANCE F

WHERE F.NAME = C.CNAME

GROUP BY C.CNAME)) T

ORDER BY SAVING DESC;

 **תוצאה**:

**זמן ביצוע**: 0.126 שניות

שאילתא מס' 2 **תיאור**: בדיקת פרטי 400 תושבים בעייתיים שלא שילמו מיסים כלל לעירייה או ששילמו יותר קנסות ממיסים.  
  
  
**קוד**:

SELECT ID, PHONE, EMAIL

FROM RESIDENT R

WHERE R.ID NOT IN (SELECT ID

FROM COLLECTION C, TAXES T

WHERE C.SID = T.SID)

OR ((SELECT SUM(AMOUNT) FROM COLLECTION CL NATURAL JOIN SOURCE NATURAL JOIN FINES WHERE CL.ID = R.ID)

-

(SELECT SUM(AMOUNT) FROM COLLECTION CL NATURAL JOIN SOURCE NATURAL JOIN TAXES WHERE CL.ID = R.ID))

> 0

ORDER BY R.ADDRESS

FETCH FIRST 400 ROWS ONLY;

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי **תוצאה**:

**זמן ביצוע**: 0.305 שניות

שאילתא מס' 3  
  
**תיאור**: בדיקת פרטי האזרחים ששלחו פניות לוועדות שקיבלו הכי הרבה פניות.  
  
**קוד**:

SELECT NAME, EMAIL, MAIL

FROM RESIDENT R

WHERE ID in (SELECT I.ID

FROM COMMITTEE C, INQUIRIES I

WHERE C.CNAME = I.COMMITNAME

AND C.CNAME IN (SELECT CNAME

FROM COMMITTEE C, INQUIRIES I

WHERE C.CNAME = I.COMMITNAME

GROUP BY CNAME

HAVING COUNT(\*) = (SELECT MAX(A)

FROM (SELECT COUNT(\*) AS A

FROM COMMITTEE C, INQUIRIES I

WHERE C.CNAME = I.COMMITNAME

GROUP BY CNAME) T)));

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי **תוצאה**:

**זמן ביצוע**: 0.297 שניות

שאילתא מס' 4  
  
**תיאור**: פרטי התושבים שקיבלו קנסות או שילמו מיסים, אך לא גם קיבלו קנסות וגם שילמו מיסים.  
  
**קוד**:

SELECT ID, NAME, ADDRESS

FROM RESIDENT

WHERE ID IN (((SELECT R.ID FROM RESIDENT R, COLLECTION C, SOURCE S, FINES F

WHERE C.ID = R.ID AND C.SID = S.SID AND F.SID = C.SID)

UNION

(SELECT R.ID FROM RESIDENT R, COLLECTION C, SOURCE S, TAXES T

WHERE C.ID = R.ID AND C.SID = S.SID AND T.SID = C.SID))

EXCEPT

(SELECT ID FROM RESIDENT R

WHERE ID IN ((SELECT ID

FROM COLLECTION C, TAXES T

WHERE C.SID = T.SID)

INTERSECT

(SELECT ID

FROM COLLECTION C, FINES F

WHERE C.SID = F.SID))));

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי **תוצאה**:

**זמן ביצוע**: 0.181 שניות

שאילתא מס' 5  
  
**תיאור**: טבלת סיכום כל סוגי המוסדות לפי התקציב שלהן, כל עוד הוועדה שאחראית על ועדה זו קיבלה יותר מ-10 פניות.  
  
**קוד**:

SELECT I.TYPE, SUM(F.AMOUNT)

FROM INSTITUTION I, COMMITTEE C, FINANCE F, INQUIRIES INQ

WHERE I.COMMITNAME = C.CNAME AND F.NAME = C.CNAME AND INQ.COMMITNAME = C.CNAME

GROUP BY I.TYPE

HAVING COUNT(INQ.IID) > 10

ORDER BY SUM(F.AMOUNT) DESC;

תמונה שמכילה טקסט, שולחן

התיאור נוצר באופן אוטומטי **תוצאה**:

**זמן ביצוע**: 0.329 שניות

שאילתא מס' 6  
  
**תיאור**: בדיקה אילו אנשים (לא בהכרח תושבי העירייה) הכניסו לעירייה הכי הרבה כספים, כל עוד ממוצע הכספים שהם הביאו לעירייה גדול מ-1000.  
  
**קוד**:

SELECT ID, SUM(AMOUNT)

FROM ((SELECT F.SID, S.AMOUNT, F.REASON AS TYPE, C.ID AS ID

FROM FINES F, COLLECTION C, SOURCE S

WHERE F.SID = C.SID AND C.SID = S.SID)

UNION ALL

(SELECT T.SID, S.AMOUNT, T.TYPE AS TYPE, C.ID AS ID

FROM TAXES T, COLLECTION C, SOURCE S

WHERE T.SID = C.SID AND C.SID = S.SID)

UNION ALL

(SELECT D.SID, S.AMOUNT, 'Donation' AS TYPE, D.DID AS ID

FROM DONATION D, SOURCE S

WHERE D.SID = S.SID)) T

GROUP BY ID

HAVING AVG(AMOUNT) > 1000;

תמונה שמכילה טקסט, שולחן

התיאור נוצר באופן אוטומטי **תוצאה**:

**זמן ביצוע**: 0.281 שניות

**עכשיו שתי שאילתות עם המון נתונים, כדי לבדוק גם שאילתות בסדרי גודל של שניות**

שאילתא מס' 7  
  
**תיאור**: בדיקת מהו אחוז התקציב של כל מוסד עבור כל שנה מסך התקציב הכללי, ממויין לפי שמות בסדר אלפביתי.  
   
**קוד**:

SELECT INS.INSTNAME, F.YEAR,   
SUM(F.AMOUNT)/(SELECT SUM(AMOUNT) FROM FINANCE WHERE YEAR = F.YEAR) AS PERCENT\_OF\_BUDGET

FROM INSTITUTION INS, FINANCE F, COMMITTEE C

WHERE INS.COMMITNAME = C.CNAME AND F.NAME = C.CNAME

GROUP BY INS.INSTNAME, F.YEAR

ORDER BY INSTNAME;

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי **תוצאה**:

**זמן ביצוע**: 13.481 שניות

שאילתא מס' 8  
  
**תיאור**: פורים מגיע, ולו נרצה לבצע משחק וגמד ענק בין כל תושבי העירייה, יש שאילתא לביצוע הגרלת זוגות אוטומטי 😊

**קוד**:

SELECT R1.NAME AS NAME1, R1.ADDRESS AS ADDRESS1, R2.NAME AS NAME2, R2.ADDRESS AS ADDRESS2

FROM (SELECT ID, NAME, ADDRESS, ROW\_NUMBER() OVER (ORDER BY id) RN FROM (SELECT \*

FROM RESIDENT

ORDER BY ID

FETCH FIRST 2500 ROWS ONLY) T) R1,

(SELECT ID, NAME, ADDRESS, ROW\_NUMBER() OVER (ORDER BY id) RN FROM (SELECT \*

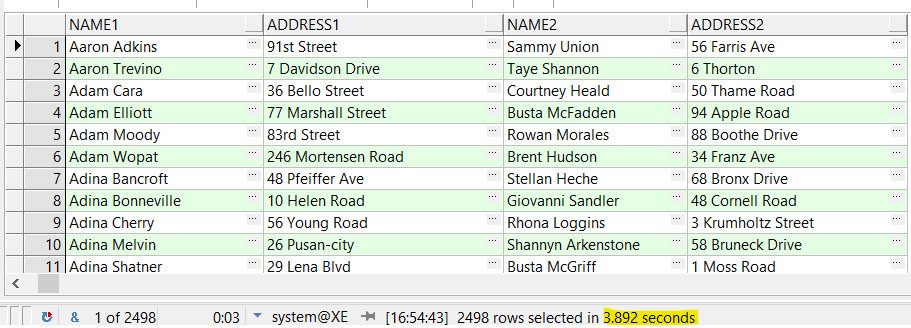
FROM RESIDENT

ORDER BY ID DESC

FETCH FIRST 2500 ROWS ONLY) T) R2

WHERE MOD(R1.RN + FLOOR(dbms\_random.value(1, 10)), (SELECT COUNT(\*) FROM RESIDENT)) = R2.RN;

ORDER BY R1.NAME;

** תוצאה**:

**זמן ביצוע**: 3.892 שניות.

## אינדקסים

1. **האינדקס**:  
   create index inst\_indx on INSTITUTION ( commitname );  
     
   **שאילתא לדוגמה שמשתמשת בו**: [שאילתא מס' 7](#שאילתא7).  
     
     
   **השוואת זמנים**:  
   תמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטי

|  |  |  |
| --- | --- | --- |
| **לפני** | **אחרי** | **שיפור?** |
| 13.481 | 11.890 | V |

**נימוק**: ניתן לראות שיפור עצום של שניה וחצי כיוון שאכן יש שימוש רב בעמודת ה-commitname בשאילתא שעליה אנחנו בודקים.

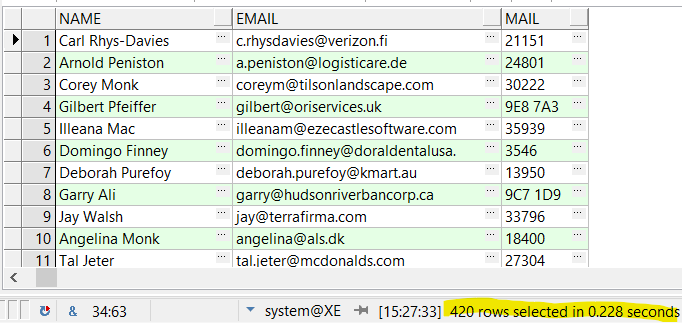
1. **האינדקס**:

create unique index clc\_indx on COLLECTION ( sid, id );  
  
  
**שאילתא לדוגמה שמשתמשת בו**: [שאילתא מס' 4](#שאילתא4).  
  
  
**השוואת זמנים**:  


|  |  |  |
| --- | --- | --- |
| **לפני** | **אחרי** | **שיפור?** |
| 0.181 | 0.131 | V |

**נימוק**: ניתן לראות שיפור בגלל שיש הרבה השוואות בשאילתא זו בין id ו-sid וכך ההשוואות מתבצעות מהר יותר.

1. **האינדקס**:

create index inq\_indx on INQUIRIES ( commitname );  
  
  
**שאילתא לדוגמה שמשתמשת בו**: [שאילתא מס' 3](#שאילתא3).  
  
  
**השוואת זמנים**:  


|  |  |  |
| --- | --- | --- |
| **לפני** | **אחרי** | **שיפור?** |
| 0.297 | 0.228 | V |

**נימוק**: ישנו שיפור בגלל שכשאנחנו מוסיפים להשוואה את commitname בבדיקה של ההשוואה של ה-Inquiries שלנו.

1. **האינדקס**:

create index inq\_indx on RESIDENT ( name );  
  
  
**שאילתא לדוגמה שמשתמשת בו**: [שאילתא מס' 8](#שאילתא8).  
  
  
**השוואת זמנים**:  


|  |  |  |
| --- | --- | --- |
| **לפני** | **אחרי** | **שיפור?** |
| 3.892 | 2.790 | V |

**נימוק**: בגלל המיון על פי name, כשאנו מבצעים אינדקס של NAME זה גורם למיון מהיר יותר, מה שבפועל הוביל שחסכנו כמעט שניה של זמן ריצה!

1. **האינדקס**:

create index inq\_indx on RESIDENT ( name );  
  
  
**שאילתא לדוגמה שמשתמשת בו**: [שאילתא מס' 5](#שאילתא5).  
  
  
**השוואת זמנים**:  
תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

|  |  |  |
| --- | --- | --- |
| **לפני** | **אחרי** | **שיפור?** |
| 0.329 | 0.337 | X |

**נימוק**: כיוון ש-name הוא כבר חלק מהמפתח כבר יש אינדקס עבור הזוג (name, year), ובנוסף כמעט ואין שורות בטבלה זו ולכן האינדקס גם כן לא משפיע כל כך.

1. **האינדקס**:

create index src\_ne\_indx on SOURCE ( amount );  
  
  
**שאילתא לדוגמה שמשתמשת בו**: [שאילתא מס' 6](#שאילתא6).  
  
  
**השוואת זמנים**:  


|  |  |  |
| --- | --- | --- |
| **לפני** | **אחרי** | **שיפור?** |
| 0.281 | 0.348 | X |

**נימוק**: בגלל שהשימוש בשדה ה-amount הוא לא על מנת להשוות או למיון, אלא סתם בשביל חישוב הוא לא יוצר שום שיפור בזמן הריצה כמובן אלא רק סרבולים מיותרים.

# חלק ג'

## דיאגרמות

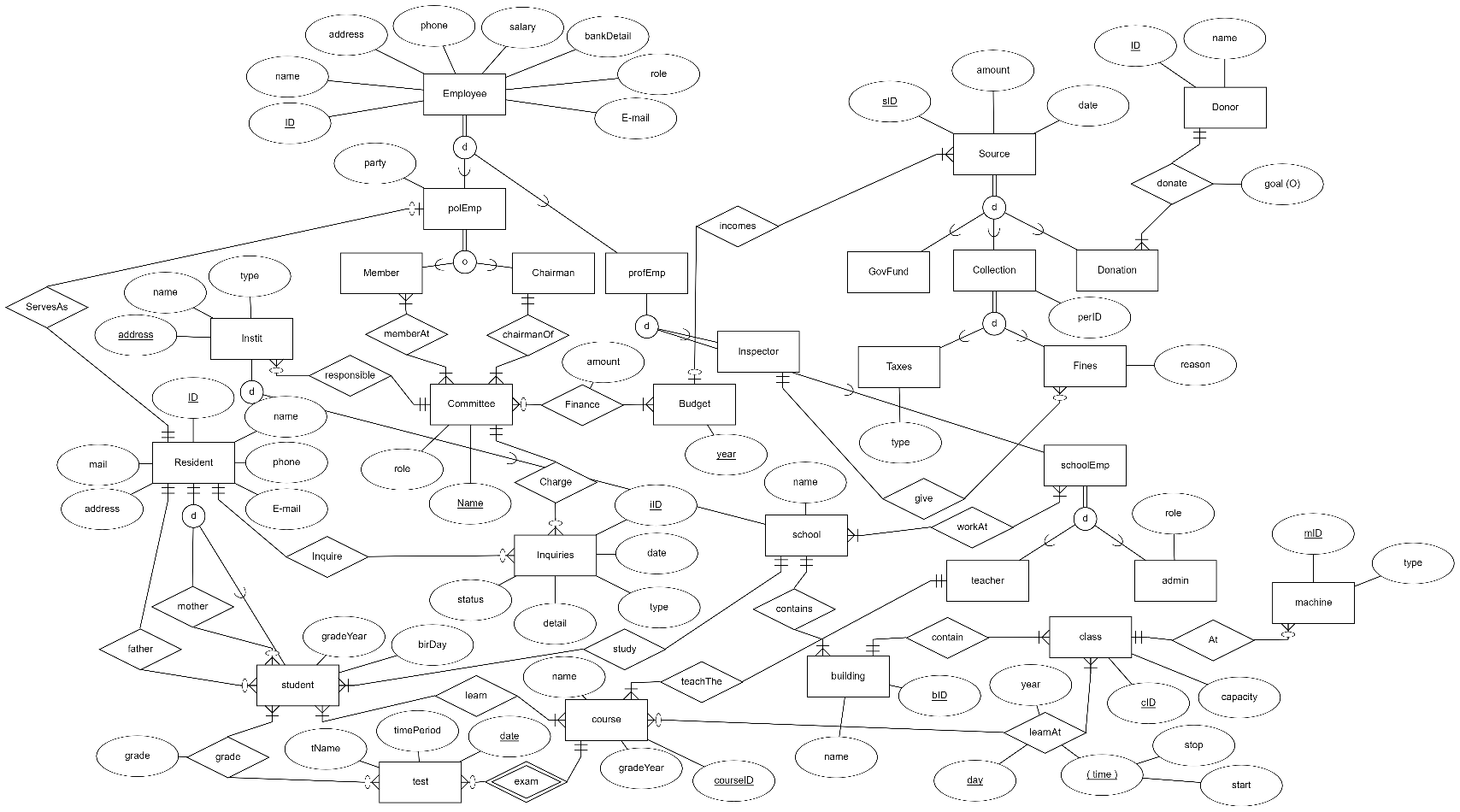
****עירייה – פרויקט 1:

בית ספר – פרויקט 2:

תמונה שמכילה טקסט, סושי

התיאור נוצר באופן אוטומטי

דיאגרמה משולבת:



## התאמות

נוסיף מספר ישויות על מנת להתאים בין המודלים:

1. **SchoolEmp**

מייצג עובד בית הספר. יורש מעובד מקצועי.

תכונות:

* ID – ת"ז (מפתח, מפתח זר של ProfEmp)

CREATE TABLE schoolEmp  
(  
 ID INT NOT NULL, תז עובד  
 PRIMARY KEY (ID),  
 FOREIGN KEY (ID) REFERENCES profEmp (ID)  
);

1. **Admin**

מייצג מנהל בית ספר. יורש מעובד בית ספר.

תכונות:

* ID – ת"ז (מפתח, מפתח זר של ProfEmp)
* Role – דרגת ניהול (מספר)

CREATE TABLE admin  
(  
 role INT NOT NULL,  
 ID INT NOT NULL,  
 PRIMARY KEY (ID),  
 FOREIGN KEY (ID) REFERENCES schoolEmp (ID)  
);  
מורה  
CREATE TABLE teacher  
(  
 ID INT NOT NULL,  
 PRIMARY KEY (ID),  
 FOREIGN KEY (ID) REFERENCES schoolEmp (ID)  
);

1. **Course**

מייצג קורס / שיעור בבית הספר.

תכונות:

* courseID – מספר מזהה של הקורס
* Name – שם הקורס
* ID – ת"ז (מפתח, מפתח זר של Teacher)
* Role – דרגת ניהול (מספר)

CREATE TABLE course  
(  
 gradeYear INT NOT NULL, שנה שלה הקורס מיועד  
 courseID INT NOT NULL check (courseID > 0), מזהה הקורס  
 name VARCHAR(10) NOT NULL, שם הקורס  
 ID INT NOT NULL, תז מורה  
 PRIMARY KEY (courseID),  
 FOREIGN KEY (ID) REFERENCES teacher (ID)  
);

1. **School**

מייצג בית ספר. יורש מ-Institution.

תכונות:

* Name – שם בית הספר
* address – כתובת (מפתח, מפתח זר של Institution)

CREATE TABLE school  
(  
 name VARCHAR(20) NOT NULL, שם בית הספר  
 address varchar(30) NOT NULL, כתובת  
 PRIMARY KEY (address),  
 FOREIGN KEY (address) REFERENCES INSTITUTION (address)  
);

**5. Building**

מייצג בניין בית ספר.

תכונות:

* Name – שם בית הספר
* address – כתובת (מפתח זר של School)
* bID – מזהה בניין (מפתח)

CREATE TABLE building  
(  
 name VARCHAR(15) NOT NULL, שם הבניין  
 bID INT NOT NULL, מספר הבניין  
 address varchar(30) NOT NULL, כתובת בית הספר של הבניין  
 PRIMARY KEY (bID),  
 FOREIGN KEY (address) REFERENCES school (address)  
);

1. **Class**

מייצג כיתה בבניין בית ספר.

תכונות:

* capacity – כמה מקומות יש בכיתה
* bid – מזהה בניין (מפתח זר של Building)
* cid – מזהה כיתה (מפתח)

CREATE TABLE class  
(  
 cID INT NOT NULL, מספר כיתה  
 capacity INT NOT NULL,מקומות בכיתה  
 bID INT NOT NULL,מספר בניין שבו הכיתה נמצאת  
 PRIMARY KEY (cID),  
 FOREIGN KEY (bID) REFERENCES building (bID)  
);

1. **Student**

מייצג תלמיד מבית הספר. יורש מתושב בעירייה.   
תכונות:

* gradeYear – הכיתה של התלמיד
* birDay – תאריך הלידה של התלמיד
* Address – כתובת בית הספר של התלמיד (מפתח זר של School)
* fatherID – ת"ז אבא (מפתח זר של Resident)
* motherID – ת"ז אמא (מפתח זר של Resident)
* ID – ת"ז (מפתח, מפתח זר של Resident)

CREATE TABLE student  
(  
 gradeYear INT NOT NULL, הכיתה של התלמיד  
 birDay DATE NOT NULL, יום הולדת  
 ID INT NOT NULL,תז  
 address varchar(30) NOT NULL,כתובת בית הספר של התלמיד  
 fatherID INT NOT NULL, תז אבא  
 motherID INT NOT NULL,תז אמא  
 PRIMARY KEY (ID),  
 FOREIGN KEY (ID) REFERENCES Resident (ID),  
 FOREIGN KEY (address) REFERENCES school (address),  
 FOREIGN KEY (fatherID) REFERENCES Resident (ID),  
 FOREIGN KEY (motherID) REFERENCES Resident (ID),  
 CONSTRAINT stud check (fatherID!=motherID),  
 CONSTRAINT stud1 check (ID!=fatherID and ID!=motherID)  
);

1. **Test**

מייצג מבחן בבית הספר. יורש מתושב בעירייה.   
תכונות:

* tDate – תאריך מבחן
* tName – שם מבחן
* timePeriod – אורך המבחן
* courseID – מזהה קורס (מפתח זר של Course)

CREATE TABLE test  
(  
 tDate DATE NOT NULL, תאריך הבחינה  
 tName VARCHAR(15) NOT NULL, שם המבחן  
 timePeriod INT NOT NULL check (timePeriod > 30), זמן הבחינה(לא פחות מחצי שעה)  
 courseID INT NOT NULL, מזהה הקורס של המבחן  
 PRIMARY KEY (tDate, courseID),  
 FOREIGN KEY (courseID) REFERENCES course (courseID)  
);

1. **WorkAt**

מייצג קשר בין עובד בית ספרי לבית ספר, על מנת לאפשר לעובד בית ספר לעבוד בכמה בתי ספר.

תכונות:

* id – ת"ז עובד (מפתח חלקי, מפתח זר של SchoolEmp)
* address – כתובת בית הספר (מפתח חלקי, מפתח זר של School)

CREATE TABLE workAt  
(  
 ID INT NOT NULL, תז עובד  
 address varchar(30) NOT NULL,מזהה בית ספר  
 PRIMARY KEY (ID, address),  
 FOREIGN KEY (ID) REFERENCES schoolEmp (ID),  
 FOREIGN KEY (address) REFERENCES school (address)  
);

1. **LearnAt**

מייצג קשר בין קורס לכיתה.

תכונות:

* day – יום השיעור (מפתח חלקי)
* timestart – שעת התחלת השיעור
* timeStop – שעת סיום השיעור
* year – השנה שבה יתקיים השיעור (מפתח חלקי)
* courseID – מספר הקורס (מפתח חלקי, מפתח זר של Course)
* cID – מזהה הכיתה שבה יתקיים השיעור (מפתח חלקי, מפתח זר של Class)

CREATE TABLE learnAt  
(  
 day INT NOT NULL check(day>0 and day<7), יום בשבוע  
 timeStart TIMESTAMP NOT NULL, שעת תחילת השיעור  
 timeStop TIMESTAMP NOT NULL,שעת סיום השיעור  
 year INT NOT NULL, שנה שבה נלמד הקורס  
 courseID INT NOT NULL,מספר הקורס  
 cID INT NOT NULL,מספר הכיתה  
 PRIMARY KEY (day,timeStart, year, cID),  
 FOREIGN KEY (courseID) REFERENCES course (courseID),  
 FOREIGN KEY (cID) REFERENCES class (cID)  
);

1. **Learn**

מייצג קשר בין תלמיד לקורס.

תכונות:

* id – ת"ז התלמיד (מפתח חלקי, מפתח זר של Student)
* courseID – מספר הקורס (מפתח חלקי, מפתח זר של Course)

CREATE TABLE learn  
(  
 ID INT NOT NULL,תז תלמיד  
 courseID INT NOT NULL,מזהה הקורס  
 PRIMARY KEY (ID, courseID),  
 FOREIGN KEY (ID) REFERENCES student (ID),  
 FOREIGN KEY (courseID) REFERENCES course (courseID)  
);

1. **Grade**

מייצג קשר בין תלמיד לקורס.

תכונות:

* id – ת"ז התלמיד (מפתח חלקי, מפתח זר של Student)
* courseID – מספר הקורס (מפתח חלקי, מפתח זר של Course)
* tDate – תאריך מבחן (מפתח חלקי, מפתח זר של Test)
* Grade – ציון במבחן

CREATE TABLE grade  
(  
 ID INT NOT NULL,תז תלמיד  
 tDate DATE NOT NULL,תאריך מבחן  
 courseID INT NOT NULL,מזהה הקורס  
 grade FLOAT NOT NULL,ציון במבחן  
 PRIMARY KEY (ID, tDate, courseID),  
 FOREIGN KEY (tDate, courseID) REFERENCES test (tDate, courseID),  
 FOREIGN KEY (ID) REFERENCES student (ID)  
);

## אילוצים – Constraints

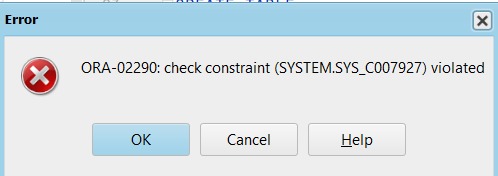
1. ניצור אילוץ שמוודא שאין שיעור (Course) שהמזהה שלו קטן מאפס (שלילי)

קוד האילוץ:

check (courseID > 0)

שאילתת בדיקת האילוץ:

insert into course values(1,-1, 'thb', 6556452);

תוצאה:

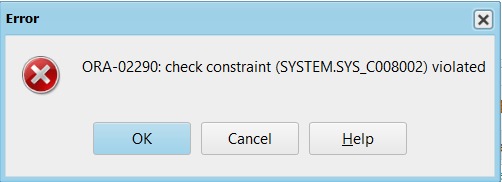
1. ניצור אילוץ שמוודא שכל מבחן אורך לפחות חצי שעה (30 דקות).

קוד האילוץ:

check (timePeriod > 30)

שאילתת בדיקת אילוץ:

insert into test values (to\_date('31/12/2021', 'dd/mm/yyyy'), 'thb', 29, 46);

תוצאה:

## שאילתות

1. ניצור view שכולל את כל המורים ואת המשכורת שהם מקבלים.  
   view זה יכול לעזור על מנת לגשת בקלות למשכורות של המורים, בנוסף לקביעת הסכם שכר כלשהו ושאר מרעין בישין שהסתדרות המורים וארגון המורים גורם להם ☹

קוד יצירת ה-view:

create or replace view teachsala as

select TEACHER.ID,salary from TEACHER join EMPLOYEE on TEACHER.ID=EMPLOYEE.ID;

תיאור שאילתא על ה-view:  
  
ניצור שאילתא שמחזירה את פרטי המורה בעל המשכורת הגבוהה ביותר ובעל המשכורת הנמוכה ביותר.  
  
קוד השאילתא:

select\*

from EMPLOYEE

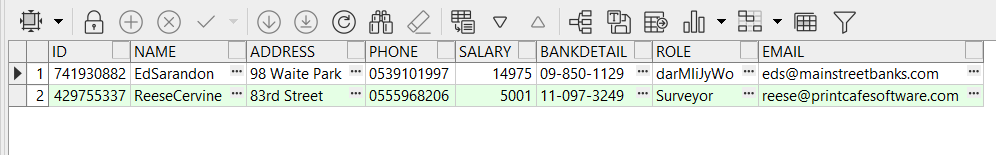
natural join (select ID

from TEACHSALA

WHERE SALARY = (SELECT MAX(SALARY) FROM TEACHSALA)

or SALARY = (SELECT min(SALARY) FROM TEACHSALA));

תוצאה:



1. ניצור view שכולל את כל התלמידים ואת פרטי האבא והאמא של כל אחד מהם.  
   view זה יכול לעזור על מנת לגשת בקלות לפרטי הורי תלמידים כאשר מתעוררת בעיה או כאשר יש צורך לפנות אליהם (או לבן שלהם, כי בתקופת הבית ספר לא מביאים פלאפון לילדים!)

קוד יצירת ה-view:

create view studdetail (studentID, studentname, fathername, mothername, phonefather, phonemother) as

select s.ID, r.name, r1.NAME, r2.name, r1.PHONE, r2.PHONE

from STUDENT s

join (select ID, name from RESIDENT) R on R.ID = s.ID

join (select ID, name, phone from RESIDENT) R1 on s.FATHERID = R1.ID

join (select Id, name, phone from RESIDENT) R2 on s.MOTHERID = R2.ID;

תיאור שאילתא על ה-view:

ניצור שאילתא שבודקת מה פרטי התלמיד שתעודת הזהות שלו היא 356642979 כי הוא התנהג מאוד לא יפה היום ובמהלך משחק כדורגל הוא הכה את חבריו.  
  
  
  
קוד השאילתא:

select phonefather,phonemother from studdetail where studentID=356642979

תוצאה:

# חלק ד'

## תצפיות – Views

1. **תיאור מילולי** – נבנה תצפית המייצגת את התקציב של כל שנה כסכום של ההכנסות של כל שנה, כך שעבור השקיפות של העירייה יהיה ניתן לראות בקלות ובבהירות מה התקציב השנתי עבור כל שנה ושנה. בנוסף, גם האחראי על התקציבים יוכל לקחת מכך השראות לסכומים של התקציבים לשנים הבאות.  
     
   **יצירת התצפית** –

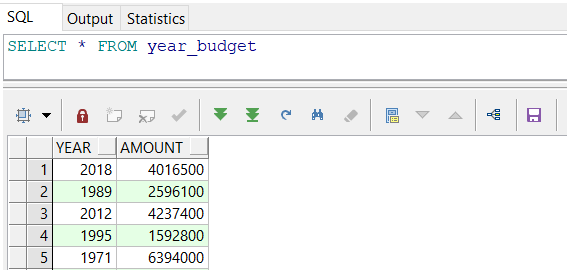
CREATE VIEW year\_budget as

SELECT B.year as YEAR, sum(amount) AS amount

FROM BUDGET B, SOURCE S

WHERE B.YEAR = S.YEAR

GROUP BY B.YEAR;

**בדיקה שאכן התצפית נוצרה והערכים נכנסו** -  
  
  
**תיאור שאילתא על התצפית** – נרצה לחפש מה התקציבים שגדולים מהממוצע של התקציבים סה"כ, על מנת ליצור השנה תקציב יותר גדול ממנו, כיוון שהשנה ההכנסות של העירייה גדלו ב"ה 😊  
  
**השאילתא על התצפית** –

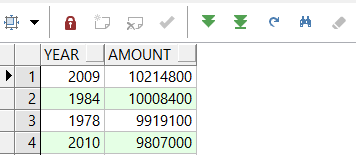
SELECT \*

FROM year\_budget

WHERE amount >= (SELECT AVG(AMOUNT)

FROM year\_budget)

ORDER BY amount DESC;

**הרצת השאילתא על התצפית -**

1. **תיאור מילולי** – נבנה תצפית המציינת מי ראשי הוועדות המוניציפליות השונים, מה שמותיהם, מה תפקידיהם ומה המייל שלהם, על מנת לאפשר לציבור הרחב לתקשר ולשלוח פניות לראש הוועדה הרלוונטי על ידי השגת פרטיו בקלות בעזרת התצפית הזאת.  
     
   **יצירת התצפית** –

CREATE VIEW chairman\_name\_role\_email AS

SELECT name, cname AS commitee\_name, email

FROM RESIDENT NATURAL JOIN COMMITTEE;

**בדיקה שאכן התצפית נוצרה והערכים נכנסו** -  
תמונה שמכילה שולחן

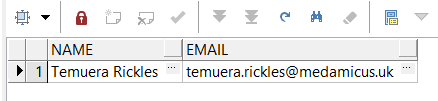
התיאור נוצר באופן אוטומטי  
  
**תיאור שאילתא על התצפית** – נרצה לבדוק מי ומה פרטיו של יו"ר הוועדה המוניציפלית שאחראית על ספורט ותרבות על מנת לשלוח למייל שלו בקשה לשפץ ולשדרג את מגרש הכדורגל הקטן והמסכן במכון.  
  
**השאילתא על התצפית** –

SELECT name, email

FROM chairman\_name\_role\_email

WHERE commitee\_name LIKE 'Sport%'

**הרצת השאילתא על התצפית -**



וגילינו בהצלחה שזהו האדון טאמורה ריקלס הנכבד.

## פונקציות – Functions

1. נגדיר פונקציה על מנת לבצע העלאה במשכורת של כמות אחוזים כלשהי לכל העובדים שאחראים על תפקיד מסוים (role). הפונקציה בנוסף תחזיר כמה תעלה ההעלאה לחודש, כלומר, מהו סכום ההעלאות הכללי על מנת שנוכל להיערך בהתאם לכך.

קוד הפונקציה:

create or replace function update\_salaries(emp\_role in VARCHAR2, percent in INTEGER) return NUMBER is

FunctionResult NUMBER;

cur\_raise number;

sum\_of\_raises number := 0;

begin

for emp in (select \*

from employee

where erole=emp\_role)

loop

cur\_raise := ceil(emp.salary \* (percent) / 100);

sum\_of\_raises := sum\_of\_raises + cur\_raise;

update employee e set e.salary = e.salary + cur\_raise

where e.id = emp.id;

end loop;

functionResult := sum\_of\_raises;

return(functionResult);

end update\_salaries;

הפונקציה עברה קומפילציה בהצלחה:



נריץ את הפונקציה:

declare

emp\_role VARCHAR2(20) := 'Engineer';

percent INTEGER := 2;

result NUMBER;

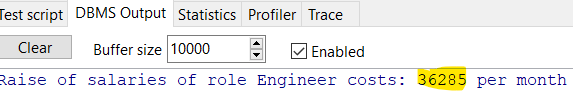
begin

result := update\_salaries(emp\_role => emp\_role, percent => percent);

DBMS\_OUTPUT.put\_line('Raise of salaries of role ' || emp\_role || ' costs: ' || :result || ' per month ');

end;

תוצאה:



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

נראה שהמשכורות התעדכנו:

1. נגדיר פונקציה המקבלת כפרמטר סכום כלשהו ובודקת האם קיים נבחר ציבור שמקבל יותר מהסכום הזה כמשכורת, ואם קיים מחזירה ‘YES’, אחרת – ‘NO’. כך התושבים בעירייה יכולים לבדוק ביעילות ובמהירות האם המשכורות בעירייה גבוהות מדי והאם צריך לצאת למחאה על כך שנבחרי הציבור שלהם בעירייה מושחתים.  
     
   קוד הפונקציה:

create or replace function check\_bigger(amount in INTEGER) return varchar2 is

cursor getSalary (eid INTEGER) is

SELECT salary

FROM EMPLOYEE

WHERE id = eid;

FunctionResult varchar2(4);

max\_salary INTEGER := 0;

cur\_salary INTEGER;

begin

for pol\_emp in (select \*

from POLEMP)

loop

OPEN getSalary(pol\_emp.id);

fetch getSalary INTO cur\_salary;

IF (max\_salary < cur\_salary) THEN

max\_salary := cur\_salary;

END IF;

CLOSE getSalary;

END LOOP;

IF (max\_salary < amount) THEN

FunctionResult := 'NO';

ELSE

FunctionResult := 'YES';

END IF;

return(FunctionResult);

end check\_bigger;

הפונקציה עברה קומפילציה בהצלחה:

תוכנית לבדיקת הפונקציה:

declare

amount INTEGER := 100000;

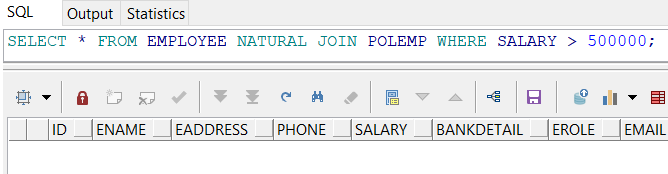
result VARCHAR2(5);

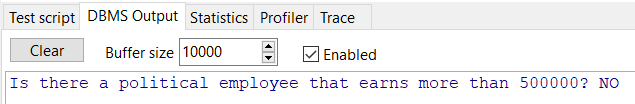
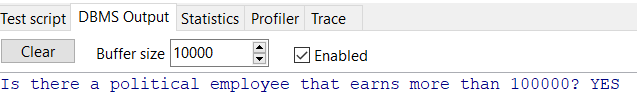
begin

result := check\_bigger(amount => amount);

DBMS\_OUTPUT.put\_line('Is there a political employee that earns more than ' || amount || '? ' || result);

end;

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיבדיקת הפונקציה:

## פרוצדורות – Procedures

1. ניצור פרוצדורה המקבלת שם של וועדה ותאריך כלשהו ומוחקת את כל הפניות שהתקבלו בוועדה שהתקבלה כפרמטר לפני התאריך שהתקבל כפרמטר וטופלו (כלומר status=’1’). פרוצדורה זו יעילה על מנת לחסוך במשאבים של נתוני העירייה שלנו.

קוד הפרוצדורה:

create or replace procedure delete\_inquries(committee\_name in out varchar2, before\_date in out Date) is

cursor getInquries(cname varchar2) IS

SELECT \*

FROM INQUIRIES

WHERE committee\_name = commitname;

inq\_id INQUIRIES.IID%type;

count\_inq INTEGER := 0;

begin

FOR inq in getInquries(committee\_name)

LOOP

IF (inq.idate < before\_date AND inq.status = '1') THEN

inq\_id := inq.iid;

DELETE FROM INQUIRIES I WHERE I.iid = inq\_id;

count\_inq := count\_inq + 1;

END IF;

END LOOP;

DBMS\_OUTPUT.put\_line('Deleted ' || count\_inq || ' inquries successfuly.');

end delete\_inquries;

הפרוצדורה עברה קומפילציה בהצלחה:



הקוד שמריץ את הפרוצדורה:

declare

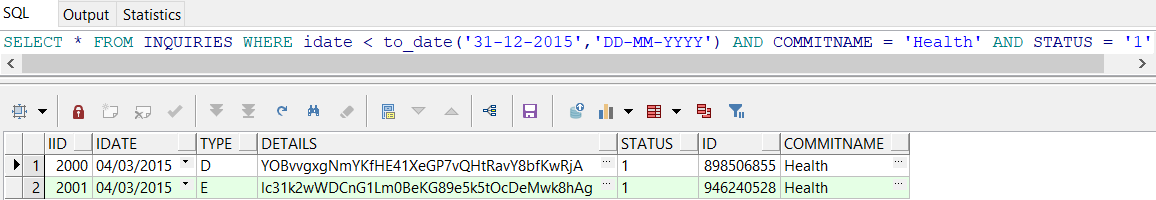
cname VARCHAR2(20) := 'Health';

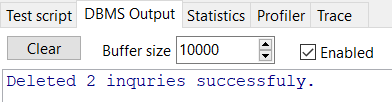
before\_date DATE := to\_date('31-12-2015','DD-MM-YYYY');

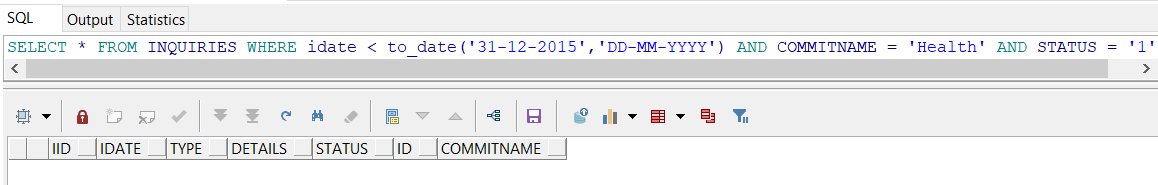
begin

delete\_inquries(committee\_name => cname, before\_date => before\_date);

end;

כלומר, נמחק את כל הפניות שהתקבלו בוועדת הבריאות לפני 31.12.2015.  
  
נבדוק שעובד – לפני ההרצה יש לנו 2 תוצאות שעומדות בתנאים:

  
ולאחריה אין כלל:



1. ניצור פרוצדורה המקבלת שם של טבלה, שם של עמודה ושם נוסף ומשנה את שם העמודה בטבלה שהתקבל כפרמטר לשם החדש. פרוצדורה זו יעילה למקרים בהם אנשים שפחות מבינים ב-SQL ובפרט ב-ORACLE SQL יוכלו בקלות ובלי סיבוכים מיותרים לשנות שמות של עמודות בטבלאות בלי שתהיה בעיה. בנוסף, בפרוצדורה זו יש בדיקת שגיאות, כלומר היא גם מדפיסה האם השינוי התבצע בהצלחה או לא.

קוד הפרוצדורה:

create or replace procedure change\_column\_name(table\_name in out varchar2, old\_name in out varchar2, new\_name in out varchar2) is

command varchar(100);

begin

command := 'alter table ';

command := command || table\_name;

command := command || ' rename column ';

command := command || old\_name;

command := command || ' to ';

command := command || new\_name;

EXECUTE IMMEDIATE command;

COMMIT;

DBMS\_OUTPUT.put\_line('Column name changed successfully!');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.put\_line('Error in chaging name.');

end change\_column\_name;

הפרוצדורה עברה קומפילציה בהצלחה:



הקוד שמריץ את הפרוצדורה:

declare

table\_name varchar(30) := 'Donor';

old\_name varchar(30) := 'id';

new\_name varchar(30) := 'donor\_id';

begin

change\_column\_name(table\_name => table\_name,

old\_name => old\_name,

new\_name => new\_name);

end;

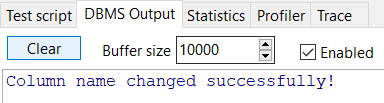
כלומר, הקוד מריץ כפרמטרים לפונקציה כך שבטבלה Donor ישתנה שם העמודה "id" ל”donor\_id”.

נבדוק שעובד:

לפני ההרצה: אחרי ההרצה:

תמונה שמכילה טקסט, שולחן

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, שולחן

התיאור נוצר באופן אוטומטי

## טריגרים – Triggers

1. ניצור טריגר שבודק האם אנו מנסים לעדכן או להוסיף עובד (דהיינו Employee) כך ששכרו נמוך משכר המינימום, על מנת שלא תהיה טעות אנוש או עבירה על החוק ח"ו ר"ל ושלא תהיה שום עוגמת נפש או צער לנו או לחברינו. במידה וזה אכן קורה הוא מעדכן את שכרו אוטומטית לשכר המינימום ומדפיס זאת למסך.

קוד הטריגר:

create or replace trigger minimum\_wage

before insert or update

on employee

for each row

declare

minimum\_wage Employee.Salary%type := 5300;

new\_emp\_salary Employee.Salary%type;

begin

new\_emp\_salary := :new.salary

IF new\_emp\_salary < minimum\_wage THEN

:new.salary := minimum\_wage;

DBMS\_OUTPUT.put\_line('Cannot pay less than minimum wage!');

END IF;

end minimum\_wage;

הטריגר עבר קומפילציה בהצלחה:

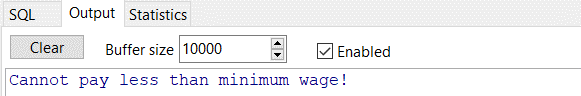


נבדוק שהוא אכן עובד – ננסה לבצע את הפקודה הבאה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

ניתן לראות שאכן בחלון ה-output מופיעה לנו הודעת השגיאה שציינו:



נבדוק שאכן שכרו של העובד שמספר הת"ז שלו הוא 374962500 אכן מקבל שכר מינימום ולא פחות מכך:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

ואכן הטריגר עבד ולא עברנו על חוק שכר מינימום, תשמ"ז-1987!

1. ניצור טריגר שיוצר ערעור (דהיינו Inquiry) אוטומטי על כל קנס (דהיינו Fine) שניתן, על מנת שהתושב בעירייה לא יצטרך לעשות זאת בעצמו, ובכך ייחסך לו ולאחרים הנוגעים בעניין זמן יקר ועוגמת נפש רבה והעיר כולה צהלה ושמחה😊

קוד הטריגר:

create or replace trigger auto\_appeal

after insert

on fines

for each row

declare

last\_iid INTEGER;

fine\_date DATE;

fined\_id INTEGER;

details VARCHAR(100);

begin

SELECT MAX(IID) INTO last\_iid FROM INQUIRIES;

SELECT sdate INTO fine\_date from SOURCE WHERE sid = :new.sid;

SELECT id INTO fined\_id FROM COLLECTION WHERE sid = :new.sid;

details := 'Appeal for fine number ' || :new.sid;

INSERT INTO INQUIRIES (IID, IDATE, TYPE, DETAILS, STATUS, ID, COMMITNAME)

VALUES (last\_iid + 1, fine\_date, 'A', details, '1', fined\_id, 'Welfare');

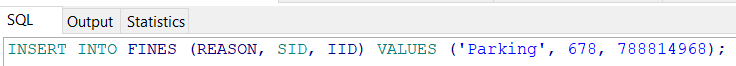
DBMS\_OUTPUT.put\_line('Automatic appeal was sent!');

end auto\_appeal;

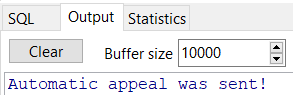
הטריגר עובר קומפילציה בהצלחה:



נבדוק שהוא אכן עובד – ננסה לבצע את הפקודה הבאה:



ניתן לראות שאכן בחלון ה-output מופיעה לנו ההודעה שציינו:



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיולבסוף נבדוק שאכן נוצר ערעור / פנייה כזאת:

ואכן הערעור נשלח בהצלחה וליהודים היתה אורה ושמחה וששון ויקר!

# חלק ה'

## גרפים

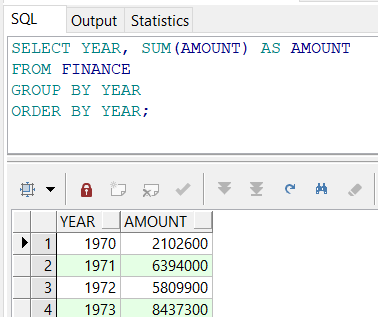
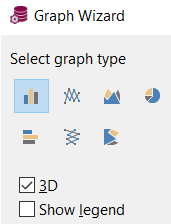
1. נריץ שאילתא הבודקת מהם סךהסכומים המוקצבים לוועדות לפי שנה, ולאחר מכן נבנה גרף מתאים.  
   שאילתא וגרף זו שימושיים על מנת לבדוק מהו המגמה של התקציב והסקה ממנה לשנים הבאות.  
     
   קוד השאילתא –

SELECT YEAR, SUM(AMOUNT) AS AMOUNT

FROM FINANCE

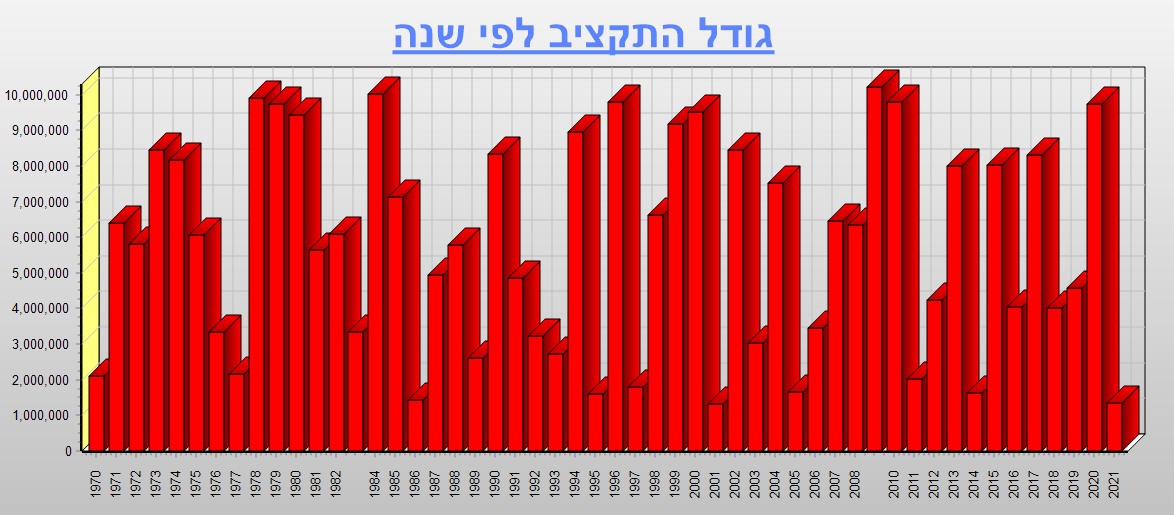
GROUP BY YEAR  
ORDER BY YEAR;

הרצת השאילתא – בחירת פרטי הגרף –

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הגרף הסופי -

1. נריץ שאילתא הבודקת מהו הסכום המוקצב לכל וועדה בשנת 2001, ולאחר מכן נבנה גרף עוגה מתאים.

שאילתא וגרף זו שימושיים על מנת לראות בצורה ויזואלית וברורה מהו החלוקה, איזו וועדה מקבלת יותר ואיזו וועדה מקבלת פחות והכל ביחס לכולל.  
  
קוד השאילתא –

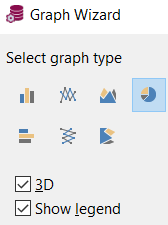
SELECT AMOUNT, NAME

FROM FINANCE

WHERE YEAR = 2001

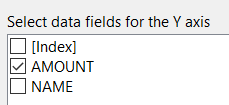
ORDER BY AMOUNT DESC;

הרצת השאילתא – בחירת פרטי הגרף –

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי



הגרף הסופי –

## דו"חות

1. נריץ שאילתא הבודקת לפי שנה המתקבלת כמשתנה (Variable) מה היו הקנסות שהתקבלו מקובצים לפי למה התקבלו, כמה התקבלו וסכום הקנסות שהתקבלו.  
   דבר זה יאפשר לנו לראות בקלות בכל שנה מה היא התפלגות הקנסות השנתית, את סכומן וכמותם ולהסיק מכך נתונים ובירורים לדברים השונים בפיקוח העירייה.

שאילתת הדו"ח –

select reason "Reason", sum(amount) "Sum", count(\*) "Count"

from fines natural join source

where year = &<name="Year"

list="select(year) from budget"

type=integer>

group by reason;

בחירת ערכי משתנים –

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטיהדו"ח שהתקבל –

1. נריץ שאילתא המקבלת וועדה כמשתנה (Variable) ומחזירה, מקובצים על פי סוג הפנייה, את כמות הפניות שהופנו אל הוועדה מאותו הסוג, ואת כמות הפניות שעוד לא טופלו.  
   דו"ח זה בעצם מאפשר הצצה מהירה, בהירה ופשוטה לקריאה של כמות הפניות שעוד לא טופלו בכל וועדה ובכך ניתן גם להשוות זאת לסך כמות הפניות הכללי כדי לקבל תמונת יחס של איזו וועדה הכי דואגת לטפל בפניות שנשלחות אליה, ובכל וועדה ניתן גם כך לדעת לאילו סוג פניות לתת עדיפות.

שאילתת הדו"ח –

select type, count(\*) "Total Count", count(CASE WHEN status='0' THEN 1 END) "Not Resolved Count"

from Inquiries I

where I.COMMITNAME = &<name="Committee"

list="select(cname) from committee"

type=string>

group by type;

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיבחירת ערכי משתנים –

הדו"ח שהתקבל –

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי