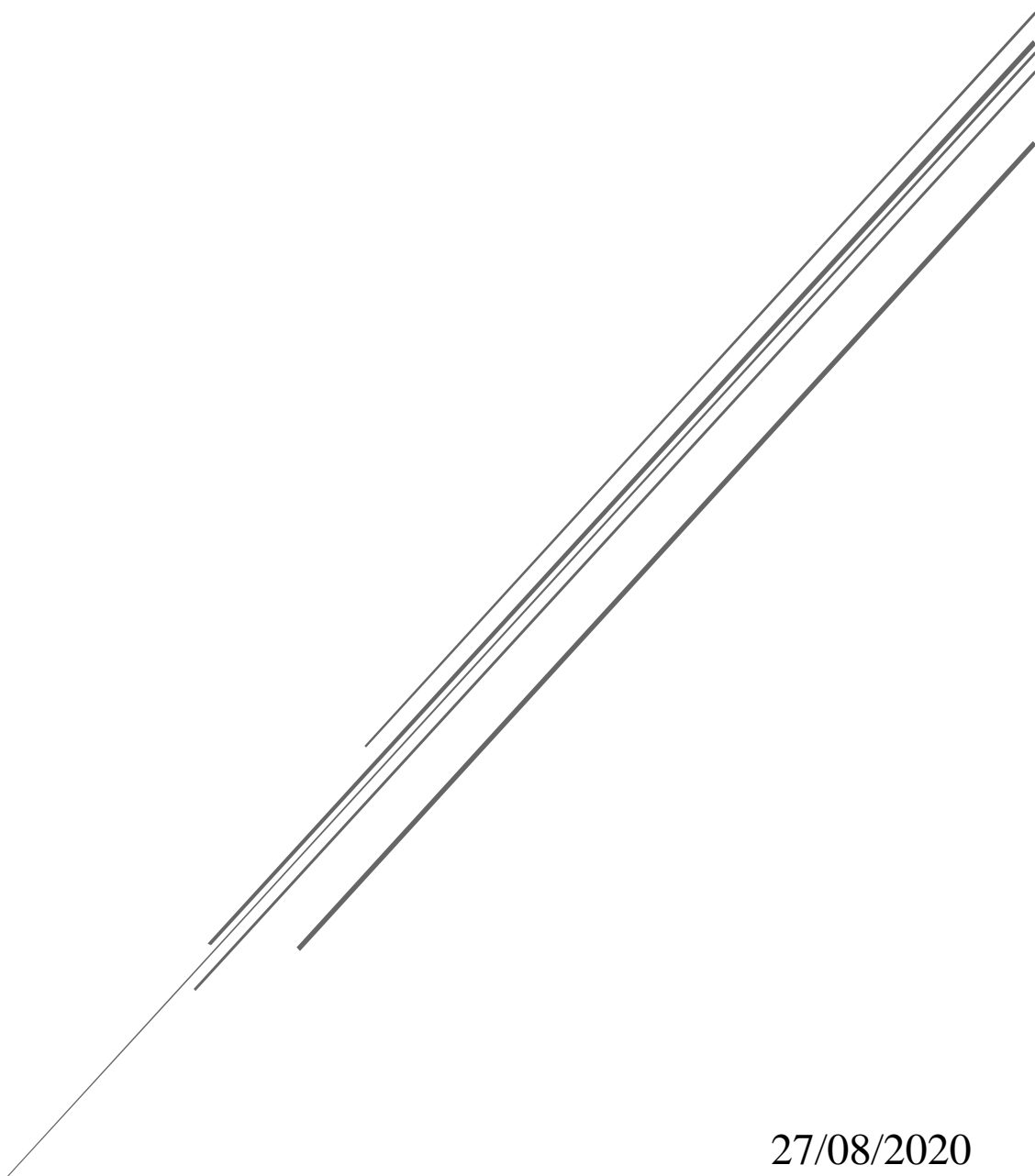


# דו"ח פרויקט

רפאל גולדיס 322680083

אסף לוי 212384507

<https://github.com/asafleve123/150225-5779-Databases.git>



27/08/2020

מיני פרויקט בבסיסי נתונים

קבוצה 49

## תוכן עניינים

2.....	מבוא – עבודת הכנה והכרת התוכנה
--------	--------------------------------

2.....	תרשים ERD
2.....	תיאור הישויות והקשרים
2 .....	ישויות
2 .....	קשרים
3 .....	נרמול הטבלאות
3.....	תרשים DSD

4.....	הפרויקט שלנו
--------	--------------

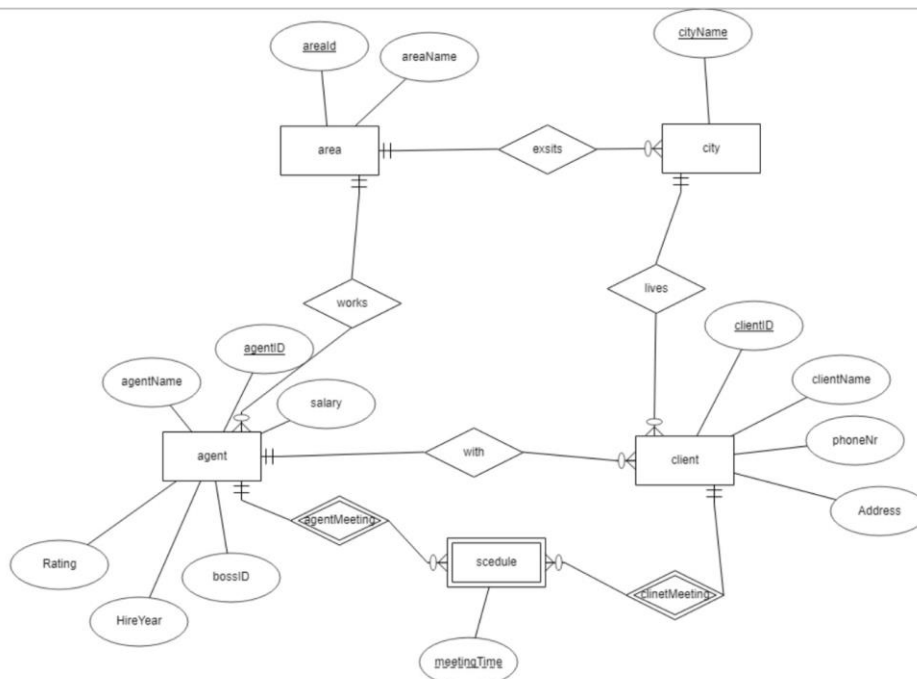
4.....	תרשים ERD
4.....	תיאור הישויות והקשרים
4 .....	ישויות
5 .....	קשרים
5.....	נרמול הטבלאות
5.....	תרשים DSD
6.....	יצירת הטבלאות
7.....	הכנסת נתונים
9.....	שאלות SQL
13 .....	אינדקסים
15 .....	אינטגרציה
15 .....	שאלות האינטגרציה
17 .....	תרשימים
18 .....	VIEWS
20 .....	פונקציות
20 .....	פרוצדורות

22.....	נספחים
---------	--------

22 .....	נספח ראשון: יצירת הטבלאות
23 .....	נספח שני: שאלות ואינדקסים
24 .....	נספח שלישי: INTEGRATION
25 .....	נספח רביעי: VIEWS
26 .....	נספח חמישי: פונקציות ופרוצדורות

## מבוא – עבודת הכנה והכרת התוכנה

### תרשים ERD



### תיאור הישויות והקשרים

#### ישויות

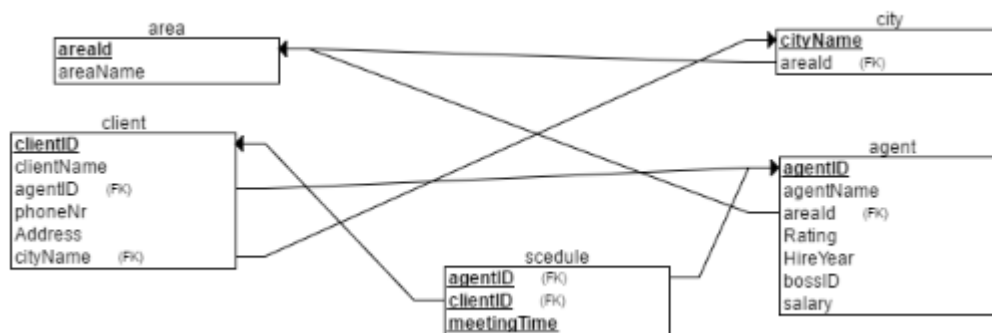
- Client - מאופיין ב: ת"ז של הלקוח, שם, כתובת, מספר פלאפון, עיר מגורים, מספר סוכן.
- Agent - מאופיין ב: ת"ז של הסוכן, שם סוכן, מספר אזור, דירוג, ותק, ת.ז של הבוס, משכורת.
- Area - מאופיין ב: מספר אזור, שם האזור.
- City - מאופיין ב: מספר אזור, שם העיר.
- Schedule - מאופיין ב: ת"ז של הסוכן, ת"ז של הלקוח, זמן הפגישה.

#### קשרים

- לכל סוכן – יש אזור עבודה יחיד, יכולים להיות לו הרבה לקוחות ויכולים להיות לו הרבה פגישות עם לקוחות.
- לכל אזור – יכולים להיות כמה ערים ויכולים להיות הרבה סוכנים שעובדים באזור זה.
- לכל עיר – יכולים להיות הרבה לקוחות שגרים בה והיא יכולה להיות באזור יחיד.
- לכל לוח זמנים – חייב להיות סוכן יחיד ולקוח יחיד הנפגשים בזמן מסוים.
- לכל לקוח – יש סוכן יחיד והוא גר בעיר יחידה ויכול להיות לו הרבה פגישות עם הסוכן שלו.

## נרמול הטבלאות

- Schedule (agentID, clientID, meetingTime)
- Agent (agentID, rating, hireYear, bossID, salary, areaID)
- Client (clientID, clientName, address, phoneNr, cityName)
- CityName (cityName, areaID)
- Area (areaID, areaName)

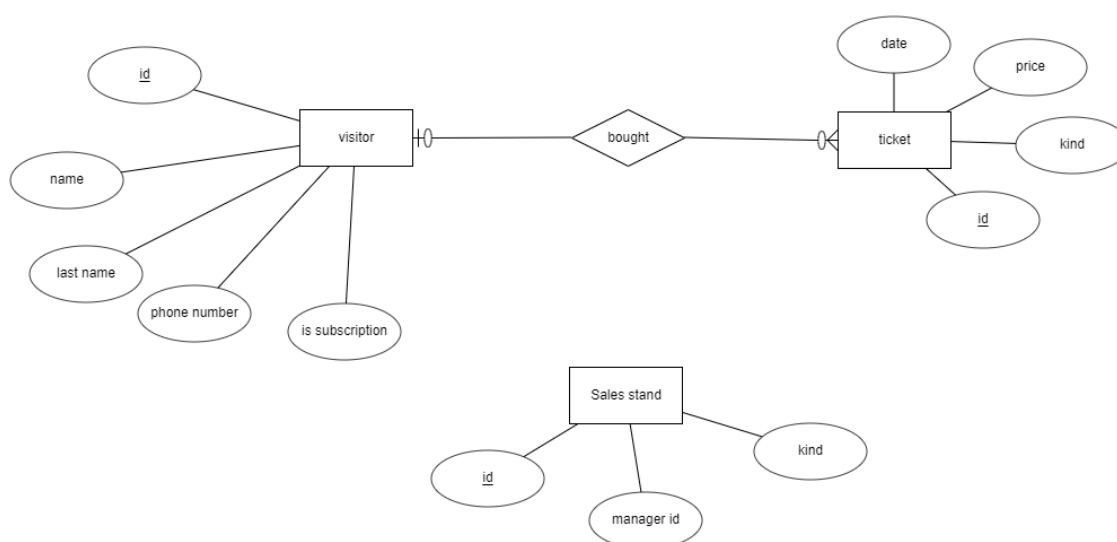
**DSD** תרשים

## הפרויקט שלנו

הקבוצה שלנו ניהלה בסיס נתונים עבור גן חיות מסוים. בסיס הנתונים יאפשר ניהול נכון של משאבי הגן וניתוח הנתונים להתייעלות והתנהגות בהתאם. החלק שלנו בתוך הקבוצה התרכז בניהול המכירות – כרטיסים ושאר מוצרים.

### תרשים ERD

בחלק שלנו ישנן 3 ישויות: מבקר בגן החיות; כרטיס; דוכן מכירה (לא של כרטיסים). בשלב הראשון יצרנו תרשים ERD שיתאר את הקשרים בין הישויות הללו ואת התכונות שלהן.



### תיאור הישויות והקשרים

#### ישויות

1. Visitor – ישות המייצגת מבקר בגן החיות.
  - Id – מספר תעודת הזהות של המבקר (PK).
  - Name – שמו הפרטי.
  - Last name – שם המשפחה.
  - Phone number – מספר הטלפון הנייד שלו.
  - Is subscription – האם הוא מנוי לגן החיות.

2. Ticket – ישות המייצגת כרטיס כניסה שנרכש.
  - Id – מספר מזהה של הכרטיס (PK).
  - Kind – סוג הכרטיס (למשל: רגיל או VIP).
  - Price – מחיר הכרטיס.
  - Date – תאריך הנפקת הכרטיס.

3. Sales stand – ישות המייצגת דוכן מכירות הנמצא בשטח גן החיות.

- Id – מספר מזהה של הדוכן (PK).
- Manager id – מספר תעודת הזהות של האחראי על הדוכן.
- Kind – סוג הדוכן (כגון מזכרות, אוכל וכדומה).

## קשרים

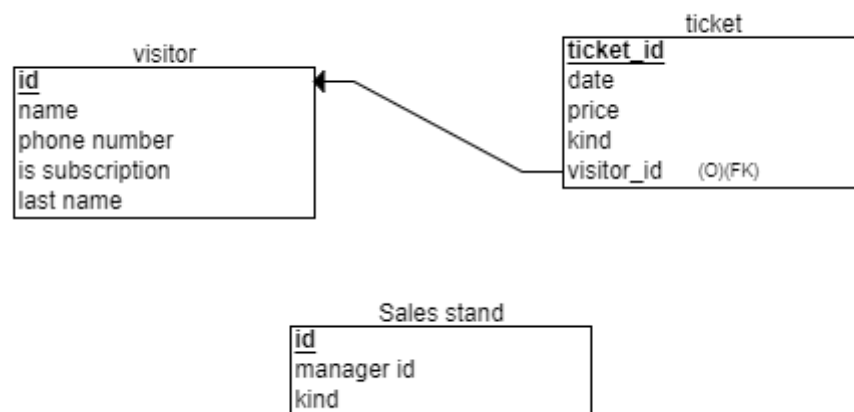
4. Bought – הקשר בין visitor לבין ticket. הקשר הוא יחיד לרבים (בדיוק אחד), משום שמבקר אחד יכול לקנות הרבה כרטיסים, אבל לכל כרטיס יש רק מבקר אחד שרכש אותו.

## נרמול הטבלאות

היחסים עומדים ב-3NF וב-BCNF : מכיוון שבכל טבלה, התלויות הפונקציונאליות הלא-טריוויאליות הן מהמפתח אל תכונות נוספות לכן מתקיים שלכל  $X, X \rightarrow Y$ ,  $X$  הוא מפתח, ולכן הם עומדים בתנאים.

## תרשים DSD

על פי תרשים ה-ERD ועל ידי הבנת הקשרים בין הישויות, יצרנו תרשים DSD עבור החלק שלנו במערכת. לא נוצרו לנו טבלאות עבור קשרים, כיוון שהקשר היחידי שיש לנו הוא קשר של "בדיוק אחד", שעבורו כידוע, לא נוצרת טבלה בפני עצמה.



## יצירת הטבלאות

אחרי שהבנו כיצד בסיס הנתונים צריך להראות בצורה מדויקת, מה תכיל כל טבלה ומהם הקשרים בין כל הטבלאות, פנינו ליצירת הטבלאות בפועל בעזרת פקודות ה-create table.

הנפקנו קוד לייצור הטבלאות באמצעות export SQL של האתר erdPlus, יצרנו קובץ SQL ואז העתקנו את קוד ה-SQL של כל טבלה אל תוכנת ה-plsql לשם יצירת הטבלאות בפועל.

```
CREATE TABLE visitor
(
  id NUMERIC(9) NOT NULL,
  name VARCHAR(20) NOT NULL,
  phone_number VARCHAR(20) NOT NULL,
  is_subscription VARCHAR(5) NOT NULL,
  last_name VARCHAR(20) NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE ticket
(
  date DATE NOT NULL,
  price INT NOT NULL,
  kind VARCHAR(20) NOT NULL,
  ticket_id NUMERIC(10) NOT NULL,
  visitor_id NUMERIC(9)
  PRIMARY KEY (ticket_id),
  FOREIGN KEY (visitor_id) REFERENCES visitor(id)
);

CREATE TABLE Sales_stand
(
  manager_id NUMERIC(9) NOT NULL,
  id NUMERIC(10) NOT NULL,
  kind VARCHAR(20) NOT NULL,
  PRIMARY KEY (id)
);
```

## הכנסת נתונים

השתמשנו בכמה כלים על מנת לאכלס את הטבלאות. הכלי הראשון היה data generator בתוכנת ה-PL/SQL, כדי לאכלס את ticket ו-sales stand. הכלי הזה מאפשר יצירה מהירה של נתונים רבים ולייצרם בעזרת סכמות קיימות. היתרון בו שהוא מובנה בתוכנה.

הכלי השני שהשתמשנו בו הוא האתר mockaroo, כדי לאכלס את visitor. האתר מאפשר בחירה מגוונת של נתוני סרק. באתר אפשר להוריד את הנתונים כקובץ של פקודות INSERT או כקובץ csv עד 1000 נתונים (בגרסה החינמית). בשל המגבלה הזו יצרנו כ-20 קבוצות קטנות של 1000 נתונים בקבצי csv, אבל אז כדי לשמור על הייחודיות של המספר המזהה (משום שאמנם כל קבוצה קטנה הייתה חוקית, אבל ביחד לא), מיזגנו אותם בעזרת אקסל (הכלי השלישי) לקבוצה אחת גדולה וחוקית.

על מנת לפתור את הבעיה שכשרוצים להכניס נתונים לכל אחד מהטבלאות ישנם ערכים שהם UNIQUE – כלומר צריכים לדאוג שהנתונים בערך הזה לא יחזרו על עצמם – מימשנו אותם כ"מספר רץ", דהיינו מספר סדרתי שעולה עם כל נתון שאנחנו מכניסים, וכך לכל נתון יש ערך ייחודי באותו שדה.

כך בוצעה יצירת והכנסת נתונים של TICKET:

TICKET					
Owner	Table	Number of records			
ASLEVI	TICKET	20000			
Name	Type	Size	Data	Master	
DATE_	DATE		Random(1/1/2010, 1/1/2021)	...	...
PRICE	NUMBER		List(30, 60, 90)	...	...
KIND	VARCHAR2	20	List('basic', 'vip')	...	...
TICKET_ID	NUMBER	10	Sequence(1)	...	...
VISITOR_ID	NUMBER	9	List(select ID from VISITOR)	...	...
*				...	...

כך בוצעה יצירת והכנסת הנתונים של SALES\_STAND: בהתחלה יצרנו רק 100 שורות, אך לאחר מכן הוספנו עוד 400 שורות. (לא יכולנו למחוק/לעדכן את ה-100 הראשונים וליצור מחדש 500 באותה התבנית, מכיוון שהזוג שהשתמש בטבלאות שלנו יצר תלות בינם לבין הטבלאות שלו).

SALES_STAND					
Owner	Table	Number of records			
ASLEVI	SALES_STAND	100			
Name	Type	Size	Data	Master	
MANAGER_ID	NUMBER	9	Sequence(1)	...	...
ID	NUMBER	10	Sequence(1)	...	...
KIND	VARCHAR2	20	List('food', 'snacks', 'animals food', 'gift')	...	...
*				...	...

SALES_STAND					
Owner	Table	Number of records			
ASLEVI	SALES_STAND	400			
Name	Type	Size	Data	Master	
MANAGER_ID	NUMBER	9	Random(101, 900)	...	...
ID	NUMBER	10	Sequence(101)	...	...
KIND	VARCHAR2	20	List('food', 'snacks', 'animals food', 'gifts')	...	...
*				...	...



כך בוצעה יצירת הנתונים בעזרת mockaroo והכנסתם באמצעות text importer:

mockaroo SCHEMAS 1 DATASETS 1 SCENARIOS APIS PROJECTS

visitor Save Changes

Field Name	Type	Options
id	Row Number	blank: 0 % <i>fx</i> ×
first_name	First Name	blank: 0 % <i>fx</i> ×
last_name	Last Name	blank: 0 % <i>fx</i> ×
phone	Phone	format: ###-###-#### blank: 0 % <i>fx</i> ×
is_subscribed	Boolean	blank: 0 % <i>fx</i> ×

Add another field

# Rows: 1000 Format: CSV Line Ending: Unix (LF) Include: ☒ header ☐ BOM

Download Data Preview Create API... More ▾

Append data: visitor ▾

---

Data from Textfile Data to Oracle

**General**

Owner: ASLEVI Table: VISITOR

Commit every... 100

☒ Overwrite duplicates ☐ Delete records

☐ Ignore duplicates ☐ Truncate table

Initializing Script

Finalizing Script

**Fields**

Field1 id -> ID

Field2 first\_name -> NAME (VARCHAR2)

Field3 last\_name -> LAST\_NAME

Field4 phone -> PHONE\_NUMBER

Field5 is\_subscribed -> IS\_SUBSCRIPT

Field ID

Fieldtype Number

Create SQL

SQL function

additional Oracle processing, for example: substr(, 1, 20)

**Result Preview**

id	first_name	last_name	phone	is_subscribed
1	Jody	Grigsby	819-319-2474	TRUE
2	Lorri	Piquard	755-328-5038	TRUE
3	Rickert	Pietersma	149-541-8354	TRUE
4	Sher	Leindecker	280-376-5451	TRUE
5	Penrod	Milliken	702-464-7698	TRUE
6	Clementina	Pattinson	680-204-3453	FALSE
7	Bartie	Kinforth	609-475-2708	TRUE

Import Import to Script Close aslevi@labdbwin visitor (19).csv loaded, 780 KB Help

## שאלות SQL

לאחר שיצרנו את בסיס הנתונים והכנסנו לתוכו מידע, כתבנו כמה שאלות מעניינות על מנת לתשאל אותן.

1. על מנת לעדכן את כלל המנויים בהטבות ופעילויות חדשות בגן החיות, כתבנו שאלתה שמחזירה את כל המבקרים שהם מנויים ובתוך כך את מספר תעודת הזהות שלהם, את שמם המלא ומספר הפלאפון שלהם.

```
SELECT ID, NAME, LAST_NAME, PHONE_NUMBER FROM VISITOR
WHERE IS_SUBSCRIPTION = 'TRUE';
```

	ID	NAME	LAST_NAME	PHONE_NUMBER
1	514	Gracia	Deverick	264-950-6718
2	515	Teodoor	Montfort	653-288-5008
3	516	Aristotle	Schimann	424-558-3348
4	517	Warren	Kenderdine	384-894-3262
5	522	Geri	Ahlin	246-348-1024
6	524	Jarvis	Conyer	561-442-2664
7	525	Christyna	Jickles	396-580-1071
8	526	Karyn	Keepe	692-335-4420
9	527	Amitie	Francisco	993-324-8480
10	528	Vonnie	Maloney	132-435-6096
11	530	Ebba	Roskrug	916-359-7696
12	532	Dieter	Gossage	207-315-8495
13	538	Felicle	Brinkworth	870-162-4972
14	539	Benjamin	Micheu	154-452-3389
15	540	Reba	Rodda	372-102-0875
16	543	Audry	Charnock	346-732-0165
17	546	Michaela	Hansford	870-707-9938
18	547	Brendin	Frith	374-495-4562
19	549	Reade	Andrejevic	396-275-3013
20	550	Daveen	Kingsnorth	717-418-3664

2. כדי לקבל תמונת מצב על דוכני המכירות שיש בגן, אנו רוצים לדעת כמה דוכנים קיימים מכל סוג.

```
select count(*), KIND from SALES_STAND
GROUP BY KIND
```

	COUNT(*)	KIND
1	113	gifts
2	112	food
3	145	snacks
4	130	animals food

3. על מנת להעריך רווחים, נרצה לדעת כמה כסף הכניס כל סוג כרטיס.

The screenshot shows the SQL Developer interface with the following query in the SQL tab:

```
select SUM(PRICE),KIND from TICKET GROUP BY KIND
```

The Output tab displays the results of the query:

	SUM(PRICE)	KIND
1	600750	vip
2	600240	basic

The status bar at the bottom indicates: 2 rows selected in 0.014 seconds.

4. כדי להמשיך להעריך רווחים, נרצה לדעת כמה כסף הכניסו כל סוג מבקרים, ומהו ממוצע ההכנסות מהסוג הזה.

The screenshot shows the SQL Developer interface with the following query in the SQL tab:

```
SELECT IS_SUBSCRIPTION,AVG(PRICE),SUM(PRICE)
FROM VISITOR JOIN TICKET ON VISITOR.ID = VISITOR_ID
GROUP BY IS_SUBSCRIPTION
```

The Output tab displays the results of the query:

	IS_SUBSCRIPTION	AVG(PRICE)	SUM(PRICE)
1	TRUE	59.875088595511	589590
2	FALSE	60.2186545848518	611400

The status bar at the bottom indicates: 2 rows selected in 0.024 seconds.


5. ההנהלה רוצה לארגן הגרלה בקרב המבקרים. כל מבקר יקבל כרטיסי הגרלה כמספר כרטיסי הכניסה שקנה. לשם כך כתבנו שאילתה שתחזיר עבור כל לקוח כמה כרטיסים קנה יחד עם פרטיו האישיים (התוצאה תמוין לפי מספר הכרטיסים בסדר יורד בשביל הנוחות).

SQL


Output

Statistics


```
SELECT ID,AMOUNT,NAME, LAST_NAME,PHONE_NUMBER
FROM (SELECT COUNT(*) AS AMOUNT, VISITOR_ID AS ID FROM TICKET GROUP BY VISITOR_ID)
NATURAL JOIN VISITOR
ORDER BY AMOUNT DESC
```



	ID	AMOUNT	NAME	LAST_NAME	PHONE_NUMBER
▶	1	7997	7 Gayle	Stiegar	975-668-0532
	2	4497	6 Portia	Lapworth	285-611-0003
	3	6873	6 Jacky	Oriel	229-979-5033
	4	7902	6 Baron	Tomaselli	846-158-2323
	5	15612	6 Aldrich	Pouck	368-613-5936
	6	8972	6 Evangelia	Huggan	192-343-9850
	7	10333	6 Wylie	Brayfield	669-331-2313
	8	13837	6 Rosemary	Heavens	663-601-1268
	9	8615	6 Hynda	Crannage	598-941-3514
	10	1711	5 Brynna	Linney	225-954-2373
	11	18791	5 Evania	Fidell	484-412-4201
	12	1212	5 Samson	Gaiford	196-626-2241
	13	1478	5 Zorine	Gough	999-545-7599
	14	2514	5 Sharleen	Beltzner	420-974-8998
	15	4284	5 Ignacius	Mongenot	146-210-8674
	16	5088	5 Nappy	Pateman	329-509-3512
	17	5372	5 Tuckie	Tunesi	147-497-2338
	18	6649	5 Correy	O' Clovan	416-928-8755

 4:21 0:04 aslevi@labdbwin 12732 rows selected in 4.234 seconds

6. כדי לדעת האם החלוקה הקיימת לסוגי כרטיסים משתלמת, נרצה לדעת כמה מבקרים מסוג מסוים קנו כרטיס מסוג מסוים.

SQL	Output	Statistics
<pre>SELECT 'TRUE' as IS_SUB,KIND,COUNT(*) as AMOUNT FROM VISITOR JOIN TICKET ON VISITOR.ID = VISITOR_ID WHERE IS_SUBSCRIPTION = 'TRUE' GROUP BY KIND UNION SELECT 'FALSE' as IS_SUB,KIND,COUNT(*) as AMOUNT FROM VISITOR JOIN TICKET ON VISITOR.ID = VISITOR_ID WHERE IS_SUBSCRIPTION = 'FALSE' GROUP BY KIND</pre>		
		
IS_SUB	KIND	AMOUNT
1 FALSE	basic	5023
2 FALSE	vip	5130
3 TRUE	basic	4943
4 TRUE	vip	4904

7:14 aslevi@labdbwin 4 rows selected in 0.026 seconds

```

SQL      Output  Statistics
SELECT count(*) as AMOUNT,extract(YEAR from DATE_) AS Year from TICKET
GROUP BY extract(YEAR from DATE_)
ORDER BY YEAR

```

	AMOUNT	YEAR
1	1779	2010
2	1829	2011
3	1830	2012
4	1759	2013
5	1812	2014
6	1811	2015
7	1792	2016
8	1768	2017
9	1802	2018
10	1916	2019
11	1902	2020

3 of 11      aslevi@labdbwin      11 rows selected in 0.026 seconds

	ID	NAME	LAST_NAME	PHONE_NUMBER
▶	1	514 Gracia	Deverick	264-950-6718
	2	515 Teodoor	Montfort	653-288-5008
	3	520 Nonna	Station	404-221-8420
	4	524 Jarvis	Conyer	561-442-2664
	5	529 Gearard	Mcettrick	220-759-3514
	6	530 Ebba	Roskrug	916-359-7696
	7	532 Dieter	Gossage	207-315-8495
	8	537 Ardith	Tubritt	810-153-9761
	9	539 Benjamin	Micheu	154-452-3389
	10	542 Eleanora	Cayette	510-764-2106
	11	544 Hillary	Baselio	782-257-1852
	12	545 Kathi	Dunster	332-467-4054
	13	547 Brendin	Frith	374-495-4562
	14	549 Reade	Andrejevic	396-275-3013
	15	552 Keene	Edmott	931-618-5042
	16	553 Claudio	Chevalier	458-170-5361
	17	554 Earvin	Cardoe	724-568-8775
	18	556 Kinner	Caldicot	644-070-0446

SQL Output Statistics

```
SELECT ID, NAME, LAST_NAME, PHONE_NUMBER FROM  
VISITOR NATURAL JOIN  
(SELECT ID FROM VISITOR  
MINUS  
SELECT VISITOR_ID as ID FROM TICKET)
```

7/10/2019 10:01 AM SQL Developer 19.3.0.170.0

aslevi@labdbwin 7268 rows selected in 1.912 seconds

## אינדקסים

אינדקסים עוזרים למצוא במהירות גדולה יותר נתונים שנשמרו בטבלאות בבסיס הנתונים. אפשר לדמות את האינדקסים כמו מראה מקום בספר. במקום שנקרא את כל הספר כדי למצוא את מה שאנחנו מחפשים נלך למראה מקום שיראה לנו את כל המקומות שבהם מוזכר הנושא שאנחנו מחפשים. השימוש באינדקסים יחסוך לנו זמן ויהפוך את תהליך החיפוש ליעיל יותר. מהבחינה הזו האינדקסים בטבלאות של ה-SQL זהים לאינדקס בספר.

לכן יצרנו אינדקסים (הקוד בנספח הראשון) שמקצרים את תהליך ביצוע השאילתות.

1. עבור שאילתה מספר 1, יצרנו אינדקס לטבלת VISITOR לפי העמודה is\_subscription, על אף שלא נכון לעשות אינדקס על פי עמודה זו (מכיוון שהיא בוליאנית). צפינו שאולי אינדקס זה יוריד את זמן הריצה של השאילתה, מכיוון שהיא בודקת ב-where את העמודה הנ"ל. ואכן זמן הריצה התקצר: לפני האינדקס זמן הריצה היה כ-3 שניות, ולאחריו הזמן היה כ-2 שניות, שיפור של סדר גודל של 1 שנייה, או של 30%.

SQL Output Statistics

```
SELECT ID,NAME, LAST_NAME, PHONE_NUMBER FROM VISITOR
WHERE IS_SUBSCRIPTION = 'TRUE'
```

	ID	NAME	LAST_NAME	PHONE_NUMBER
1	514	Gracia	Deverick	264-950-6718
2	515	Teodoor	Montfort	653-288-5008
3	516	Aristotle	Schimann	424-558-3348
4	517	Warren	Kenderdine	384-894-3262
5	522	Geri	Ahlin	246-348-1024
6	524	Jarvis	Conyer	561-442-2664
7	525	Christyna	Jickles	396-580-1071
8	526	Karyn	Keepe	692-335-4420
9	527	Amitie	Francisco	993-324-8480
10	528	Vonnie	Maloney	132-435-6096
11	530	Ebba	Roskrug	916-359-7696
12	532	Dieter	Gossage	207-315-8495
13	538	Felicle	Brinkworth	870-162-4972
14	539	Benjamin	Micheu	154-452-3389
15	540	Reba	Rodda	372-102-0875

2:31 0:02 aslevi@labdbwin 10025 rows selected in 2.119 seconds

```
SELECT ID,NAME, LAST_NAME, PHONE_NUMBER FROM VISITOR
WHERE IS_SUBSCRIPTION = 'TRUE';
```

	ID	NAME	LAST_NAME	PHONE_NUMBER
1	514	Gracia	Deverick	264-950-6718
2	515	Teodoor	Montfort	653-288-5008
3	516	Aristotle	Schimann	424-558-3348
4	517	Warren	Kenderdine	384-894-3262
5	522	Geri	Ahlin	246-348-1024
6	524	Jarvis	Conyer	561-442-2664
7	525	Christyna	Jickles	396-580-1071
8	526	Karyn	Keepe	692-335-4420
9	527	Amitie	Francisco	993-324-8480
10	528	Vonnie	Maloney	132-435-6096
11	530	Ebba	Roskrug	916-359-7696
12	532	Dieter	Gossage	207-315-8495
13	538	Felicle	Brinkworth	870-162-4972
14	539	Benjamin	Micheu	154-452-3389
15	540	Reba	Rodda	372-102-0875
16	543	Audry	Charnock	346-732-0165
17	546	Michaela	Hansford	870-707-9938
18	547	Brendin	Frith	374-495-4562
19	549	Reade	Andrejevic	396-275-3013
20	550	Daveen	Kingsnorth	717-418-3664

1:1 0:03 aslevi@labdbwin 10023 rows selected in 3.03


2. עבור שאילתה מספר 5, יצרנו אינדקס לטבלת TICKET לפי VISITOR\_ID. העמודה הזאת מגוונת והסיכוי לקבל ID מסוים הינו פחות מ-1%. צפינו שאינדקס זה ישפר את זמן הריצה מכיוון שבשאילתה הפנימית נעשה GROUPBY על פי VISITOR\_ID. ואכן זמן הריצה התקצר מ-4.2 ל-2.8 שניות, שיפור של 1.5 שניות או כ-33%.

SQL


Output

Statistics

SELECT ID,AMOUNT,NAME, LAST\_NAME,PHONE\_NUMBER  
FROM (SELECT COUNT(\*) AS AMOUNT, VISITOR\_ID AS ID FROM TICKET GROUP BY VISITOR\_ID)  
NATURAL JOIN VISITOR  
ORDER BY AMOUNT DESC



	ID	AMOUNT	NAME	LAST_NAME	PHONE_NUMBER
▶	1	7997	7 Gayle	Stiegar	975-668-0532
	2	4497	6 Portia	Lapworth	285-611-0003
	3	6873	6 Jacky	Oniel	229-979-5033
	4	7902	6 Baron	Tomaselli	846-158-2323
	5	15612	6 Aldrich	Pouck	368-613-5936
	6	8972	6 Evangelia	Huggan	192-343-9850
	7	10333	6 Wylie	Brayfield	669-331-2313
	8	13837	6 Rosemary	Heavens	663-601-1268
	9	8615	6 Hynnda	Crannage	598-941-3514
	10	1711	5 Brynna	Linney	225-954-2373
	11	18791	5 Evania	Fidell	484-412-4201
	12	1212	5 Samson	Gaiford	196-626-2241
	13	1478	5 Zorine	Gough	999-545-7599
	14	2514	5 Sharleen	Beltzner	420-974-8998



& 1:1

0:02 ASLEVI@labdbwin 12732 rows selected in 2.862 seconds

SQL

Output

Statistics

SELECT ID,AMOUNT,NAME, LAST\_NAME,PHONE\_NUMBER  
FROM (SELECT COUNT(\*) AS AMOUNT, VISITOR\_ID AS ID FROM TICKET GROUP BY VISITOR\_ID)  
NATURAL JOIN VISITOR  
ORDER BY AMOUNT DESC

	ID	AMOUNT	NAME	LAST_NAME	PHONE_NUMBER	
▶	1	7997	7	Gayle	Stiegar	975-668-0532
	2	4497	6	Portia	Lapworth	285-611-0003
	3	6873	6	Jacky	Oriel	229-979-5033
	4	7902	6	Baron	Tomaselli	846-158-2323
	5	15612	6	Aldrich	Pouck	368-613-5936
	6	8972	6	Evangelia	Huggan	192-343-9850
	7	10333	6	Wylie	Brayfield	669-331-2313
	8	13837	6	Rosemary	Heavens	663-601-1268
	9	8615	6	Hynda	Crannage	598-941-3514
	10	1711	5	Brynna	Linney	225-954-2373
	11	18791	5	Evania	Fidell	484-412-4201
	12	1212	5	Samson	Gaiford	196-626-2241
	13	1478	5	Zorine	Gough	999-545-7599
	14	2514	5	Sharleen	Beltzner	420-974-8998
	15	4284	5	Ignacius	Mongenot	146-210-8674
	16	5088	5	Nappy	Pateman	329-509-3512
	17	5372	5	Tuckie	Tunesi	147-497-2338
	18	6649	5	Correy	O' Clovan	416-928-8755


4:21

0:04

aslevi@labdbwin

12732 rows selected in 4.234 seconds

3. עבור שאילתה 3, יצרנו אינדקס לטבלת TICKET על פי KIND. עשינו זאת מכיוון שהשאילתה משתמשת בGROUPBY על פי עמודה זו. אבל משום שלמעשה יש רק שני ערכים אפשריים בKIND, זמן הריצה נשאר באותו סדר גודל.





















 select SUM(PRICE),KIND from ...

SQL




Output

Statistics

```
select SUM(PRICE),KIND from TICKET GROUP BY KIND
```

	SUM(PRICE)	KIND
1	600750	vip ...
2	600240	basic ...

   1:17

aslevi@labdbwin

2 rows selected in 0.014 seconds

SQL3.sql

SQL

Output

Statistics

```
select SUM(PRICE),KIND from TICKET GROUP BY KIND
```

## אינטגרציה

בשביל לדעת פרטים נוספים על מנהלי דוכני המכירה, קיבלנו הרשאות גישה לטבלאות שבבעלות אילן קניס ועידו פרידמן, כלומר לטבלאות worker, role, department, מה שאפשר לנו לגשת ולשייך את פרטי העובדים למנהלי הדוכנים שלנו.

```
grant all privileges on DEPARTMENT to aslevi;
grant all privileges on WORKER to aslevi;
grant all privileges on ROLES to aslevi;
```

בנוסף נתנו הרשאות גישה לטבלת דוכני המכירה שלנו לשריאל סיגל ושמואל גרבר.

אך, במהלך האינטגרציה עם אילן ועידו נתקלנו בבעיה: בעוד אנחנו הגדרנו את MANAGER\_ID להיות מספר החל מ1, הם הגדירו את WORKER.ID להיות מספר רץ החל מ100000000. כדי להתגבר על הבעיה, כאשר כתבנו שאילתות הוספנו לכל MANAGER\_ID את המספר 100000000 (לא יכולנו לעשות לו UPDATE מכיוון ששריאל ושמואל כבר השתמשו בטבלה שלנו, אז לכן התאמנו את עצמנו בכל שאילתה)

## שאילתות האינטגרציה

1. על מנת שנוכל לדעת את פרטי מנהלי הדוכנים, כתבנו את השאילתה הבאה שמצרפת את פרטי האישיים של המנהל כפי ששמורים בטבלה WORKER.

SQL

```
select w.id, w.first_name, w.last_name, w.seniority, w.role_id, s.id, s.kind
from kenis.worker w , sales_stand s
where s.manager_id+100000000 = w.id
order by w.id
```

	ID	FIRST_NAME	LAST_NAME	SENIORITY	ROLE_ID	ID	KIND
1	100000001	Consalve	Rowbottom	14	363	1	food
2	100000002	Tawnya	Setchell	8	538	2	animals food
3	100000003	Dag	Hyslop	40	344	3	animals food
4	100000004	Deni	Loddon	17	450	4	snacks
5	100000005	Nataniel	Bennedick	25	852	5	food
6	100000006	Kristel	Kennford	7	179	6	animals food
7	100000007	Esmeralda	Meaney	6	351	7	gifts
8	100000008	Katrine	McEwen	14	926	8	animals food
9	100000009	Pepi	Bickmore	2	715	9	food
10	100000010	Gearalt	Cheeke	30	700	10	food
11	100000011	Jedidiah	Bearward	36	777	11	animals food
12	100000012	Cully	Catcheside	10	454	12	gifts
13	100000013	Dionysus	Caygill	42	639	13	snacks
14	100000014	Chris	Forton	39	961	14	snacks

aslevi@labdbwin 500 rows selected in 0.409 seconds



2. בשביל להציע לעובדים להתמודד לתפקיד מנהלי דוכנים – נרצה להציע רק לעובדים שאינם מנהלי דוכנים שהניסיון שלהם הוא מעל ממוצע הניסיון של מנהלי הדוכנים.

The screenshot shows the SQL Developer interface. The SQL window contains the following query:

```
select * from kenis.worker
where seniority >= ALL(select AVG(w.seniority)
from kenis.worker w , sales_stand s
where s.manager_id+100000000 = w.id)
minus
select * from kenis.worker
where ID IN (select manager_id+100000000 from sales_stand)
```

The Results window displays a table with 18 rows of employee data. The status bar at the bottom indicates that 14004 rows were selected in 9.261 seconds.

	ID	SALARY	BANK_ACCOUNT	SENIORITY	ROLE_ID	BOSS_ID	FIRST_NAME	LAST_NAME
1	100000101	22302	4030010302	25	364	100000037	Randolph	Keith
2	100000102	21538	4060010506	27	617	100000015	Neely	Proom
3	100000103	34289	4090010712	31	432	100000095	Delcine	Tolhurst
4	100000106	19870	4180011342	30	282	100000054	Katrinka	Odda
5	100000107	26894	4210011556	33	520	100000105	Charmain	Egleton
6	100000117	27599	4510013806	26	46	100000099	Talbert	Ely
7	100000118	15175	4540014042	38	333	100000038	Ulises	Kiddye
8	100000119	29986	4570014280	40	170	100000116	Dido	Espinho
9	100000120	23615	4600014520	38	284	100000005	Edgardo	Darinton
10	100000127	12753	4810016256	39	860	100000084	Stevie	Dowers
11	100000130	19721	4900017030	28	755	100000102	Winona	Eymor
12	100000141	9111	5230020022	39	555	100000009	Odessa	Leahey
13	100000146	22129	5380021462	26	485	100000025	Xerxes	Choat
14	100000149	25309	5470022350	41	729	100000081	Kathryne	Kording
15	100000150	34408	5500022650	29	932	100000023	Rebekkah	Hudleston
16	100000153	12697	5590023562	22	20	100000074	Allison	Gaven
17	100000154	25826	5620023870	27	457	100000088	Jerrine	Panchin
18	100000156	27027	5680024492	26	431	100000017	Kenn	Bovse

3. בהמשך לשאלתה הקודמת, נרצה לדעת את ממוצע השכר של מנהלי הדוכנים כדי להודיע לעובדים שהצענו להם.

The screenshot shows the SQL Developer interface. The SQL window contains the following query:

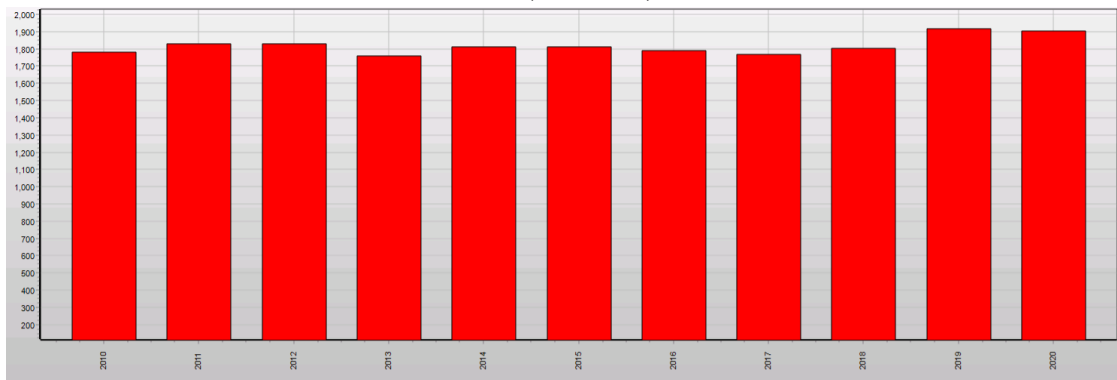
```
select AVG(w.salary)
from kenis.worker w , sales_stand s
where s.manager_id+100000000 = w.id
```

The Results window displays a single row with the average salary. The status bar at the bottom indicates that 1 row was selected in 0.023 seconds.

	AVG(W.SALARY)
1	20997.476

## תרשימים

1. תרשים המראה כמה כרטיסים נקנו בכל שנה (שאלתה 7):

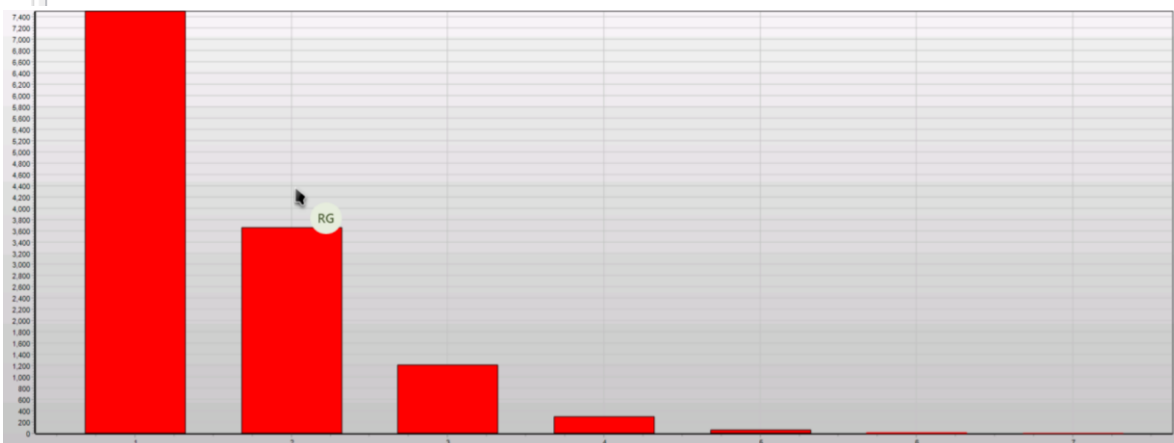


2. תרשים המראה כמה אנשים קנו כרטיס 1, כמה 2, וכו' (דומה לשאלתה 5):

SQL Output Statistics

```
SELECT AMOUNT AS NUM_OF_TICKETS, COUNT(*) AS C
FROM (SELECT COUNT(*) AS AMOUNT, VISITOR_ID AS ID FROM TICKET GROUP BY VISITOR_ID)
GROUP BY AMOUNT
```

	NUM_OF_TICKETS	C
1	1	7497
3	2	3650
6	3	1219
4	4	294
5	5	63
2	6	8
7	7	1



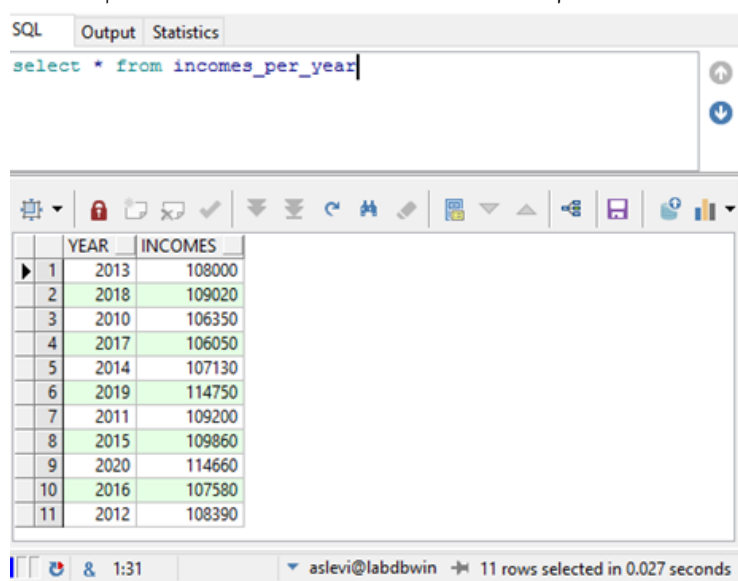
## Views

VIEWS הם טבלאות וירטואליות. VIEWS מכילים הגדרות של עמודות וסוגי מידע שאותן עמודות יכולות להכיל. ההבדל בין הטבלאות לבין ה-VIEWS הוא שבטבלאות נשמרים נתונים באופן פיזי ואילו ב-VIEWS הנתונים לא נשמרים באופן פיזי בתוכם אלא הם רק מציגים נתונים הנשמרים בטבלאות. לכן לא ניתן לעדכן או להוסיף נתונים ל-VIEWS כפי שעושים לטבלאות.

הקוד עבור יצירת ה-VIEWS בנספחים.

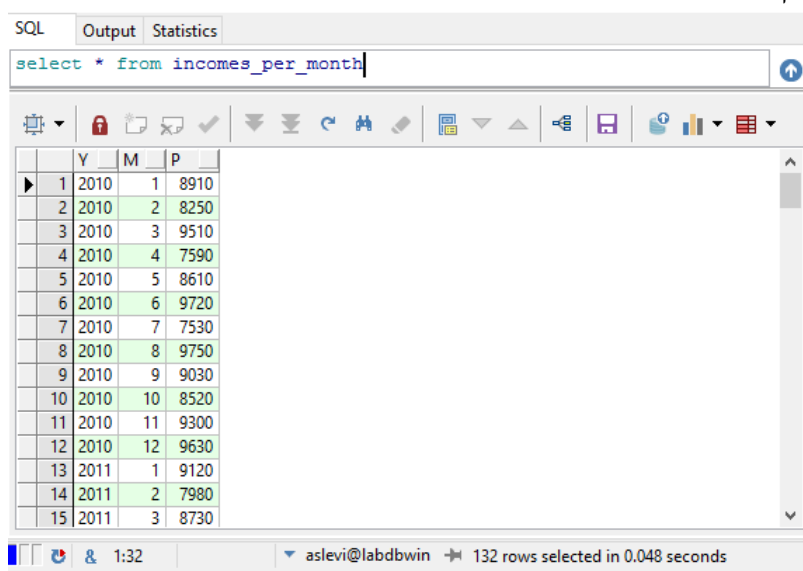
• VIEWS עבור רואה החשבון:

1. רואה החשבון של גן החיות מעוניין לצפות בהכנסות מהכרטיסים של כל שנה ושנה, על מנת שיוכל להעריך את ההכנסות לשנה הבאה ולבצע את תפקידו על הצד הטוב ביותר.



	YEAR	INCOMES
1	2013	108000
2	2018	109020
3	2010	106350
4	2017	106050
5	2014	107130
6	2019	114750
7	2011	109200
8	2015	109860
9	2020	114660
10	2016	107580
11	2012	108390

2. פעמים רבות רואה החשבון מעוניין לדעת את ההכנסות מהכרטיסים לפי חודש מסוים, לכן נאפשר לו לראות הכנסות של כל חודש בשנים האחרונות.



	Y	M	P
1	2010	1	8910
2	2010	2	8250
3	2010	3	9510
4	2010	4	7590
5	2010	5	8610
6	2010	6	9720
7	2010	7	7530
8	2010	8	9750
9	2010	9	9030
10	2010	10	8520
11	2010	11	9300
12	2010	12	9630
13	2011	1	9120
14	2011	2	7980
15	2011	3	8730

- VIEWS עבור מחלקת השיווק:

3. מחלקת השיווק של גן החיות מעוניינת לשמר את המנויים של הגן בכך שתפיץ להם דרך מספר הטלפון הטבות ומתנות.

SQL Output Statistics

```
select * from subscriptions
```

	NAME	LAST_NAME	PHONE_NUMBER
1	Gracia	Deverick	264-950-6718
2	Teodoor	Montfort	653-288-5008
3	Aristotle	Schimann	424-558-3348
4	Warren	Kenderdine	384-894-3262
5	Geri	Ahlin	246-348-1024
6	Jarvis	Conyer	561-442-2664
7	Christyna	Jickles	396-580-1071
8	Karyn	Keepe	692-335-4420
9	Amitie	Francisco	993-324-8480
10	Vonnie	Maloney	132-435-6096
11	Ebba	Roskrug	916-359-7696
12	Dieter	Gossage	207-315-8495
13	Felicle	Brinkworth	870-162-4972
14	Benjamin	Micheu	154-452-3389
15	Reba	Rodda	372-102-0875

1:28 0:01 aslevi@labdbwin 10025 rows selected in 1.605 seconds

4. מחלקת השיווק מעוניינת לגרום ללקוחותיה הלא מנויים להצטרף כמנוי לגן החיות בכך שתשלח להם הצעות מינוי (כמובן שניצור VIEW חדש פעם בכמה זמן).

SQL Output Statistics

```
select * from not_subscriptions
```

	NAME	LAST_NAME	PHONE_NUMBER
1	Erda	Moral	472-404-7217
2	Alphonse	Scrafton	682-456-3990
3	Nonna	Station	404-221-8420
4	Don	Scopham	998-517-1921
5	Ewart	Follin	347-257-3815
6	Gearard	Mcettrick	220-759-3514
7	Lottie	Dragge	712-611-9840
8	Gabriel	Mariot	265-353-3303
9	Lilith	Tackle	861-875-5101
10	Almeria	Simanek	314-624-2375
11	Rodney	Davall	747-293-3164
12	Ardith	Tubritt	810-153-9761
13	Halsey	Trott	420-148-4853
14	Eleanora	Cayette	510-764-2106
15	Hillary	Baselio	782-257-1852

1:19 0:01 aslevi@labdbwin 9979 rows selected in 1.640 seconds

## פונקציות

יצרנו 2 פונקציות שונות שמשקפות שני שימושים של פרוצדורה:

1. TICKETS\_IN\_YEAR – הפונקציה מקבלת שנה (integer) ומחזירה את מספר הכרטיסים שנמכרו באותה השנה (integer). אם הקלט שגוי, הפונקציה תחזיר 0.

```

1 begin
2   -- Call the function
3   :result := tickets_in_year(year => :year);
4 end;
```

Variable	Type	Value
result	Integer	1759
year	Integer	2013

2. Manager\_of\_sales\_stand – הפונקציה מקבלת מספר זיהוי של דוכן מכירות (integer) ומחזירה את המזהה של מנהל הדוכן (integer). אם הקלט שגוי הפונקציה תחזיר 0.

```

1 begin
2   -- Call the function
3   :result := manager_of_sales_stand(sales_stand_id => :sales_stand_id);
4 end;
```

Variable	Type	Value
result	Integer	498
sales_stand_id	Integer	101

## פרוצדורות

יצרנו 2 פרוצדורות שונות שמשקפות שני שימושים של פרוצדורה:

- Update\_to\_sub – הפרוצדורה מקבלת כקלט מספר זיהוי של מבקר (integer) ומעדכנת את השדה is\_subscription ל-TRUE – כלומר להיות מנוי.

```
1 begin
2     -- Call the procedure
3     update_to_sub(vid => :vid);
4 end;
```

<input type="checkbox"/>	Variable	Type	Value
<input checked="" type="checkbox"/>	vid	Float	300000
<input checked="" type="checkbox"/>			

```
select * from visitor where name = 'asaf';
```

	ID	NAME	PHONE_NUMBER	IS_SUBSCRIPTION	LAST_NAME
▶ 1	300000	asaf	333-333-333	TRUE	levi
2	300001	asaf	333333333	FALSE	levi
3	300002	asaf	333333333	FALSE	levi

- Add\_visitor – הפרוצדורה מוסיפה מבקר חדש (מס' זיהוי, שם, פלאפון, שם משפחה, מינוי) לתוך הסכמה של VISITOR. אם היא מצליחה להוסיף לתוך הסכמה, היא מכניסה לתוך משתנה בוליאני added אמת; אם היא לא מצליחה (לדוג': קיים בסכמה אותו מס' זיהוי) היא מכניסה false ומדפיסה הודעת שגיאה למסך ה-DBMS output:  
 מצב הצלחה: ○

```

1 declare
2   -- Boolean parameters are translated from/to integers:
3   -- 0/1/null <--> false/true/null
4   added boolean;
5 begin
6   -- Call the procedure
7   add_visitor(id => :id,
8               name => :name,
9               phone => :phone,
10              sub => :sub,
11              last_name => :last_name,
12              added => added);
13   -- Convert false/true/null to 0/1/null
14   :added := sys.diutil.bool_to_int(added);
15 end;
```

Variable	Type	Value
id	Integer	300000
name	String	asaf
phone	String	333-333-333
sub	String	FALSE
last_name	String	levi
added	Integer	1

```
select * from visitor where name = 'asaf' |
```

ID	NAME	PHONE_NUMBER	IS_SUBSCRIPTION	LAST_NAME
1 300000	asaf	333-333-333	FALSE	levi
2 300001	asaf	333333333	FALSE	levi
3 300002	asaf	333333333	FALSE	levi

- מצב תקלה(המבקר כבר קיים במאגר):

```

1 declare
2   -- Boolean parameters are translated from/to integers:
3   -- 0/1/null <--> false/true/null
4   added boolean;
5 begin
6   -- Call the procedure
7   add_visitor(id => :id,
8               name => :name,
9               phone => :phone,
10              sub => :sub,
11              last_name => :last_name,
12              added => added);
13   -- Convert false/true/null to 0/1/null
14   :added := sys.diutil.bool_to_int(added);
15 end;
```

Variable	Type	Value
id	Float	1
name	String	asdasd
phone	String	adsd
sub	String	asda
last_name	String	asdasd
added	Integer	0

## נספחים

### נספח ראשון: יצירת הטבלאות

```
CREATE TABLE visitor
```

```
(  
  id NUMERIC(9) NOT NULL,  
  name VARCHAR(20) NOT NULL,  
  phone_number VARCHAR(20) NOT NULL,  
  is_subscription VARCHAR(5) NOT NULL,  
  last_name VARCHAR(20) NOT NULL,  
  PRIMARY KEY (id)  
);
```

```
CREATE TABLE ticket
```

```
(  
  date DATE NOT NULL,  
  price INT NOT NULL,  
  kind VARCHAR(20) NOT NULL,  
  ticket_id NUMERIC(10) NOT NULL,  
  visitor_id NUMERIC(9)  
  PRIMARY KEY (ticket_id),  
  FOREIGN KEY (visitor_id) REFERENCES visitor(id)  
);
```

```
CREATE TABLE Sales_stand
```

```
(  
  manager_id NUMERIC(9) NOT NULL,  
  id NUMERIC(10) NOT NULL,  
  kind VARCHAR(20) NOT NULL,  
  PRIMARY KEY (id)  
);
```

**נספח שני: שאלות ואינדקסים**

1. SELECT ID,NAME,LAST\_NAME,PHONE\_NUMBER FROM VISITOR  
WHERE IS\_SUBSCRIPTION = 'TRUE'
  2. select count(\*),KIND from SALES\_STAND  
GROUP BY KIND
  3. select SUM(PRICE),KIND from TICKET  
GROUP BY KIND
  4. SELECT IS\_SUBSCRIPTION,AVG(PRICE),SUM(PRICE)  
FROM VISITOR JOIN TICKET ON VISITOR.ID = VISITOR\_ID  
GROUP BY IS\_SUBSCRIPTION
  5. SELECT ID,AMOUNT,NAME,LAST\_NAME,PHONE\_NUMBER  
FROM (SELECT COUNT(\*) AS AMOUNT, VISITOR\_ID AS ID FROM TICKET  
GROUP BY VISITOR\_ID) NATURAL JOIN VISITOR  
ORDER BY AMOUNT DESC
  6. SELECT 'TRUE' as IS\_SUB,KIND,COUNT(\*) as AMOUNT FROM VISITOR  
JOIN TICKET ON VISITOR.ID = VISITOR\_ID  
WHERE IS\_SUBSCRIPTION = 'TRUE'  
GROUP BY KIND  
UNION  
SELECT 'FALSE' as IS\_SUB,KIND,COUNT(\*) as AMOUNT FROM VISITOR  
JOIN TICKET ON VISITOR.ID =VISITOR\_ID  
WHERE IS\_SUBSCRIPTION = 'FALSE'  
GROUP BY KIND
  7. SELECT count(\*) as AMOUNT, extract(YEAR from DATE\_) AS Year  
from TICKET  
GROUP BY extract(YEAR from DATE\_)  
ORDER BY YEAR
  8. SELECT ID,NAME,LAST\_NAME,PHONE\_NUMBER  
FROM VISITOR NATURAL JOIN  
(SELECT ID FROM VISITOR  
MINUS  
SELECT VISITOR\_ID as ID FROM TICKET)
- 
1. create index ind\_sub on VISITOR(IS\_SUBSCRIPTION);  
[drop index ind\_sub;]
  2. create index ind\_vis on TICKET(VISITOR\_ID);  
[drop index ind\_vis;]
  3. create index ind\_kind on TICKET(KIND);  
[drop index ind\_kind;]



## נספח שלישי: Integration

### פקודות grant:

```
grant all privileges on DEPARTMENT to aslevi;  
grant all privileges on WORKER to aslevi;  
grant all privileges on ROLES to aslevi;
```

### שאלות אינטגרציה:

1. 

```
select w.id,w.first_name, w.last_name, w.seniority, w.role_id, s.id,s.kind  
from kenis.worker w, sales_stand s  
where s.manager_id+100000000 = w.id  
order by w.id
```
2. 

```
select * from kenis.worker  
where seniority >= ALL(select AVG(w.seniority)  
from kenis.worker w,sales_stand s  
where s.manager_id+100000000 = w.id)  
minus  
select * from kenis.worker  
where ID IN (select manager_id+100000000 from sales_stand)
```
3. 

```
select AVG(w.salary)  
from kenis.worker w, sales_stand s  
where s.manager_id+100000000 = w.id
```

**נספח רביעי: views**

1. create view incomes\_per\_year as  
select extract(year from DATE\_) as Year,sum(price) as incomes from ticket  
group by extract(year from DATE\_)
2. create view incomes\_per\_month as  
select extract(year from date\_) as y, extract(month from date\_) as m,sum(price) as p  
from ticket  
group by extract(month from date\_), extract(year from date\_)  
order by y,m
3. create view subscriptions as  
select name,last\_name,phone\_number from visitor  
where is\_subscription = 'TRUE'
4. create view not\_subscriptions as  
select name,last\_name,phone\_number from visitor  
where is\_subscription = 'FALSE'

**נספח חמישי: פונקציות ופרוצדורות**

1. create or replace function tickets\_in\_year(year in integer) return integer is  
 FunctionResult integer;  
 begin  
   select count(\*) into FunctionResult from ticket where extract(year from DATE\_) = year;  
   return(FunctionResult);  
 end tickets\_in\_year;  
 /
  
2. create or replace function manager\_of\_sales\_stand(sales\_stand\_id in integer) return integer is  
 FunctionResult integer;  
 begin  
   select manager\_id into FunctionResult from sales\_stand where id = sales\_stand\_id;  
   return(FunctionResult);  
 end manager\_of\_sales\_stand;  
 /
  
1. create or replace procedure add\_visitor(id in integer,name in string, phone in string, sub in string,last\_name in string,added out boolean) is  
 begin  
   INSERT INTO visitor VALUES (id, name, phone,sub,last\_name);  
   added := TRUE;  
   EXCEPTION WHEN OTHERS THEN dbms\_output.put\_line('ERROR: input was invalid or unique constraint violated');  
   added := FALSE;  
 end add\_visitor;  
 /
  
2. create or replace procedure update\_to\_sub(vid in integer) is  
 begin  
   update visitor set is\_subscription = 'TRUE' where id = vid;  
 end update\_to\_sub;  
 /