**בס"ד**

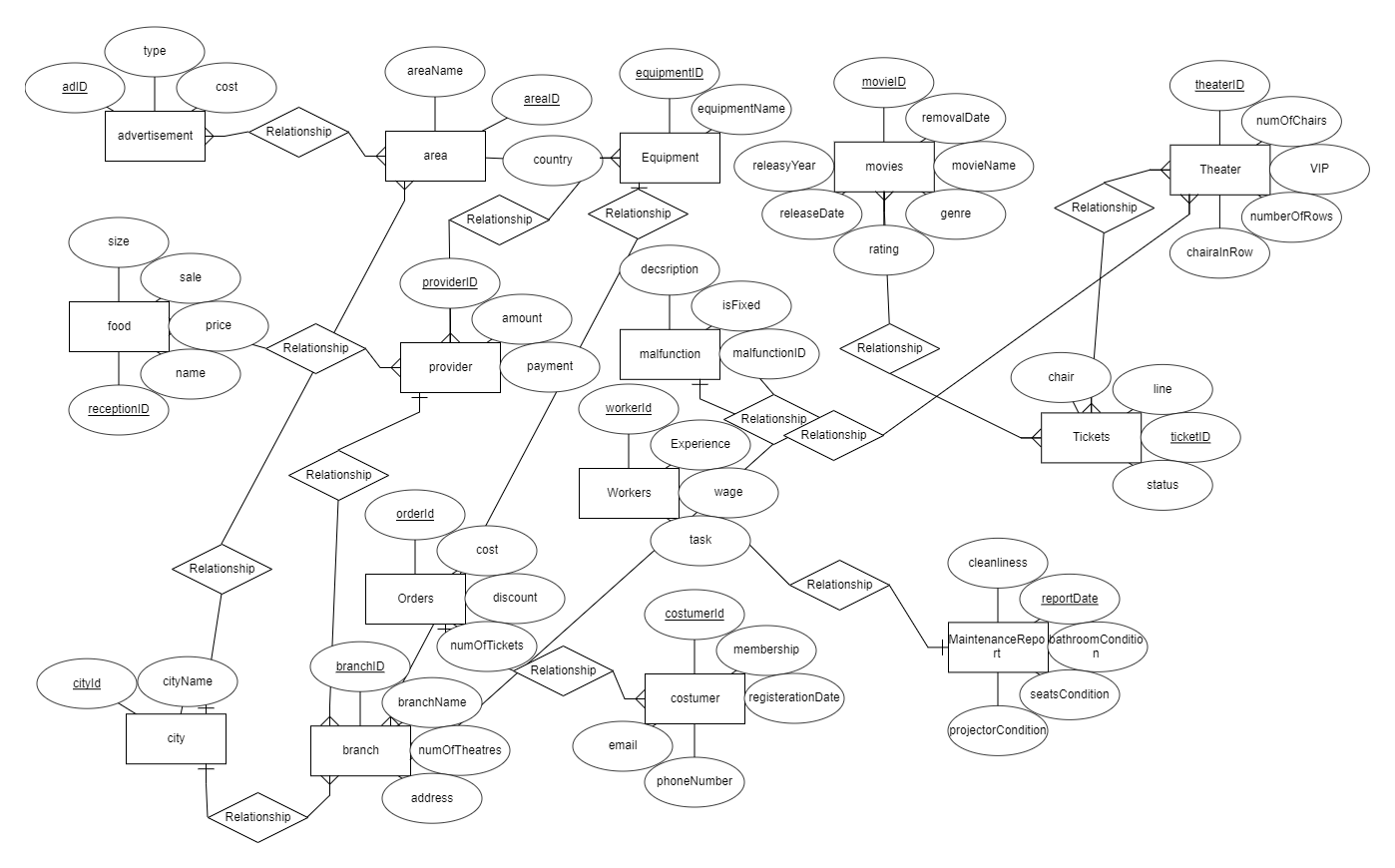
**מיני פרויקט בבסיסי נתונים**

**הוד לוי 211516562**

**אליאב ברוך 302379375**

# עבודת הכנה והכרת התוכנה

## תרשים ERD



## תיאור הישויות והקשרים

### ישויות

* Branch - מאופיין ב- מזהה של הסניף, שם, כתובת, מספר אולמות, מזהה של העיר, מזהה של הציוד ומזהה של הספק.
* Theatre - מאופיין ב- מזהה של האולם, מספר כסאות באולם, VIP או לא, מספר שורות, מספר כסאות בשורה.
* Area - מאופיין ב- מספר אזור ובשם האזור ושם המדינה.
* City מאופיין ב- מספר העיר ובשם העיר ומספר אזור.

### קשרים

* לכל סניף- יש ספק יחידי, יכול להיות לו מספר מסוים של תקלות, ומספר מסוים של ציוד ומספר מסוים של אולמות ועיר יחידה.
* לכל אולם- יכול להיות מספר מסוים של כרטיסים ומספר מסוים של תקלות.
* לכל עיר- יכולים להיות הרבה אולמות והיא יכולה להיות באזור יחידי.
* לכל אזור – יכולות להיות כמה ערים וכמה פרסומות.

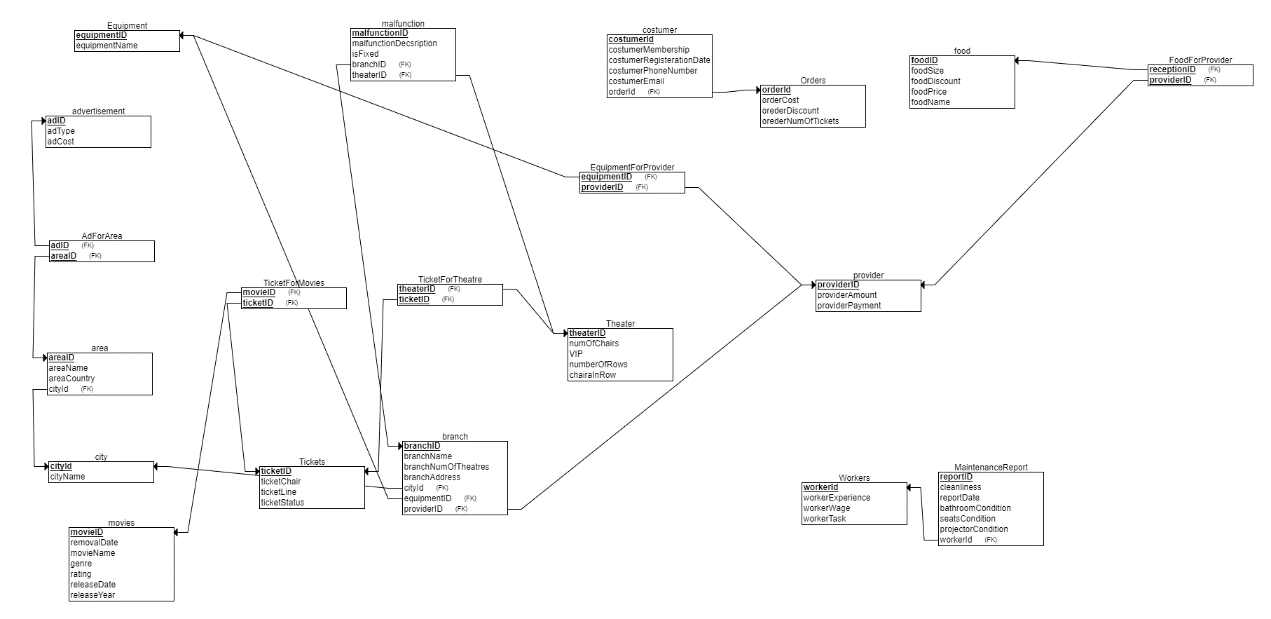
### נרמול הטבלאות

* Branch (branchID, branchName, branchNumOfTheatres, BranchAddress, cityID, equipmentID, providerID)
* Theatre (theatreID, numOfChairs, VIP, numberOfRows, chairInARow)
* CityName (cityID, cityName, areaID)
* Area (areaID, areaName, areaCountry)

### פרוקים

היחסים עומדים ב- NF3 וב- BCNF : מכיוון שבכל טבלה, התלויות הפונקציונאליות הלא-טריוויאליות הן מהמפתח אל תכונות נוספות לכן מתקיים שלכל X Y , X הוא מפתח ולכן הם עומד בתנאים.

## תרשים DSD

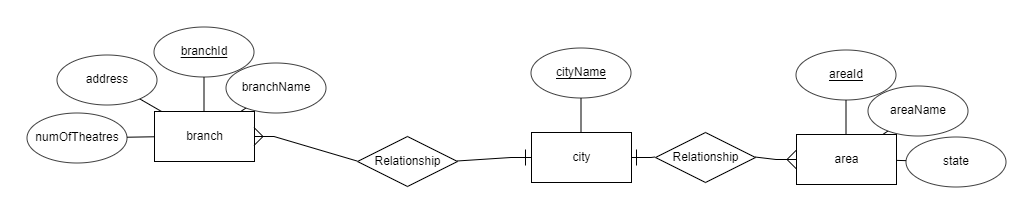
****כאן ניתן לראות את ה- DSD שהפקנו מתרשים ה- ERD שיצרנו.

# הפרויקט שלנו

*אנחנו התמקדנו ביחסים בין אלומות, סניפים, ערים ואיזורים.*

## תרשים ERD

כפי שהזכרנו, במחלקה שלנו, ישנן 3 ישויות: מתקנים, חשבונות ובקשות אספקה. בשלב הראשון יצרנו תרשים ERD שיתאר את הקשרים בין הישויות הללו ואת התכונות שלהן.



## תיאור הישויות והקשרים

### ישויות

* Branch – ישות זאת אחראית על כל המתקנים הנמצאים במערכת.

ישות זאת הינה חזקה, כיוון שיכולה להתקיים ללא תלות בישות אחרת.

* Branch\_ID – מספר מזהה של הסניף (PK)
* Branch\_name – שם הסניף
* Address – כתובת הסניף
* Num\_Of\_Theatres – מספר אולמות בסניף
* City\_ID – מספר מזהה של העיר (FK)
* City – ישות זאת אחראית על כל בקשות האספקה הנמצאים במערכת.
* ישות זאת הינה חזקה, כיוון שיכולה להתקיים ללא תלות בישות אחרת. (נעשו שינויים ממה שמוצג למעלה)
* City\_id – מספר מזהה של העיר (PK)
* City\_Name – שם העיר
* Area\_id – מספר מזהה של האיזור (FK)
* Area – ישות זאת אחראית על כל החשבונות הנמצאים במערכת.
* ישות זאת הינה חזקה, כיוון שיכולה להתקיים ללא תלות בישות אחרת.
* Area\_id – מספר מזהה של האיזור (PK)
* Area\_name – שם האיזור
* Country – מדינה

### קשרים

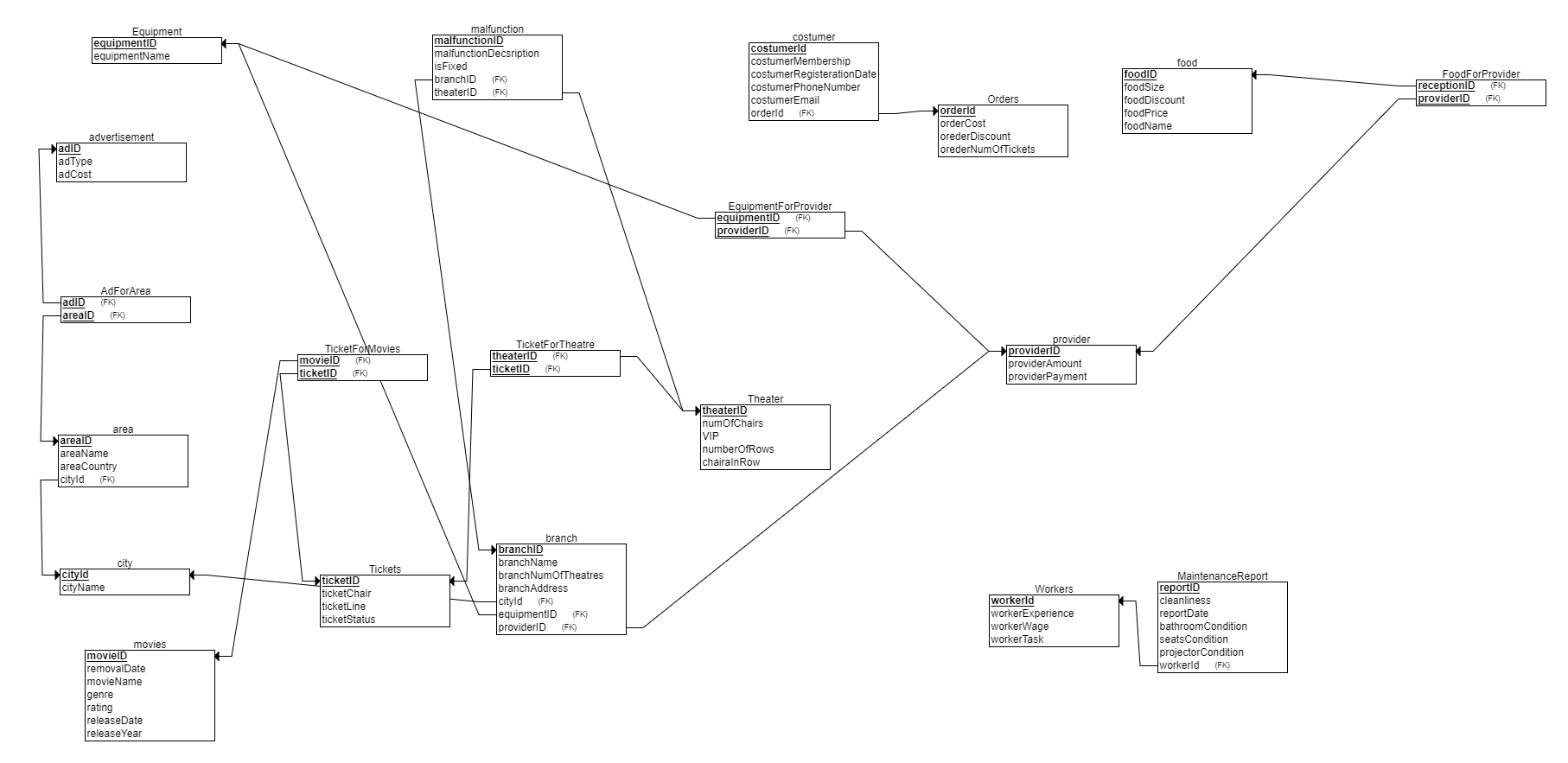
* BranchCity – הקשר בין branch לבין city. הקשר הוא M:1 משום שיכול להיות לעיר אחת הרבה סניפים, אבל סניף אחד שייך לעיר אחת.
* CityArea – הקשר בין city לבין area. הקשר הוא M:1 משום שיכול להיות לאיזור אחד הרבה ערים, אבל עיר אחת שייכת לאיזור אחד בלבד.

## נרמול הטבלאות

היחסים עומדים ב- NF3 וב- BCNF : מכיוון שבכל טבלה, התלויות הפונקציונאליות הלא-טריוויאליות הן מהמפתח אל תכונות נוספות לכן מתקיים שלכל X Y , X הוא מפתח ולכן הם עומד בתנאים.

## תרשים DSD

על פי תרשים ה- ERD ועל ידי הבנת הקשרים בין הישויות, יצרנו תרשים DSD עבור החלק שלנו במערכת: סניפים ואולמות.



## יצירת הטבלאות

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

יצרנו 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. כל הקודים נמצאים ב GITשל אורי ארליך.

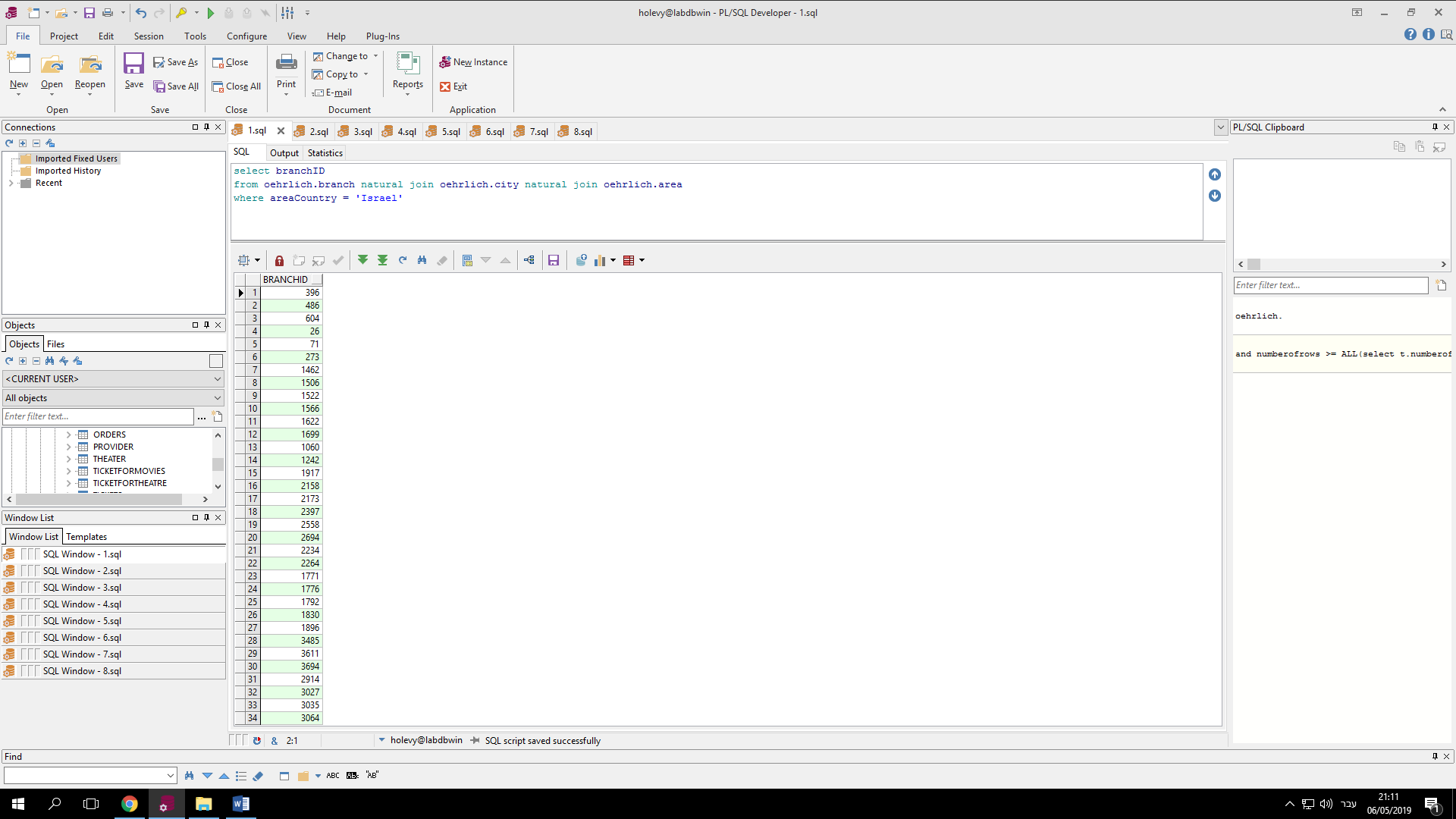
## הכנסת נתונים

השתמשנו בData generator של הPLSQL. קבצים בGIT של אורי ארליך.

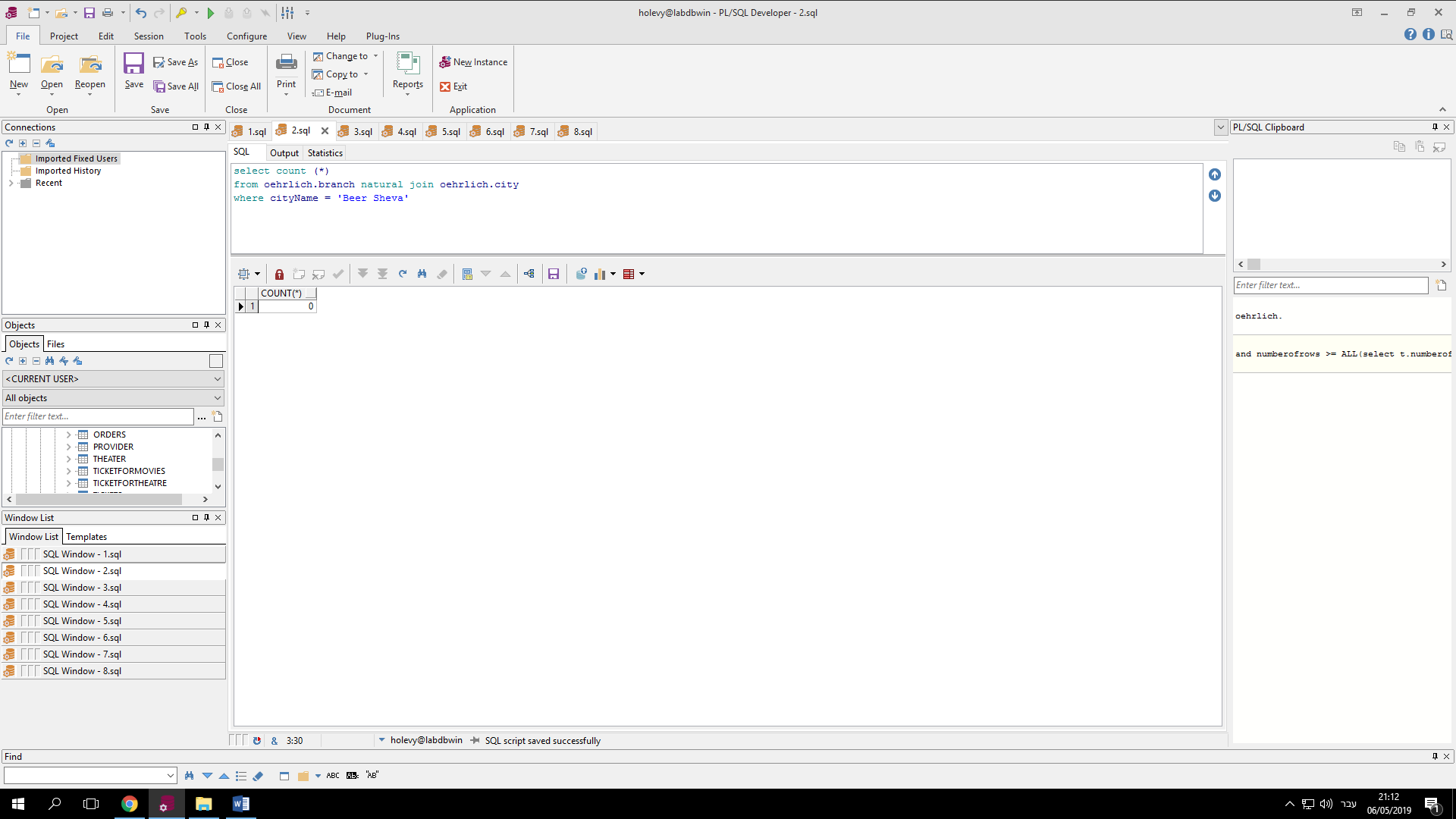
## שאילתות SQL

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

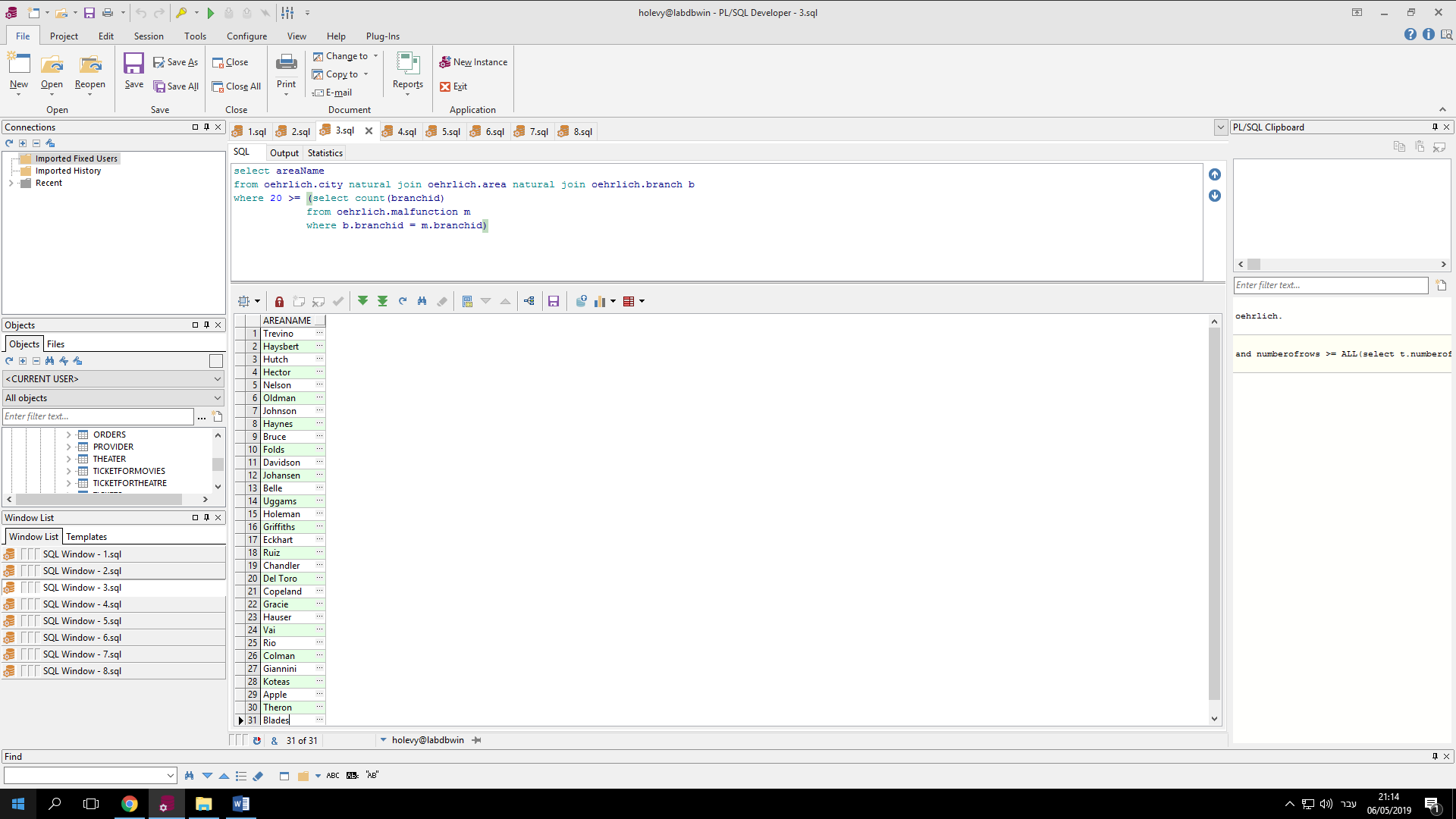
* + 1. בדקנו איזה סניפים יש בישראל

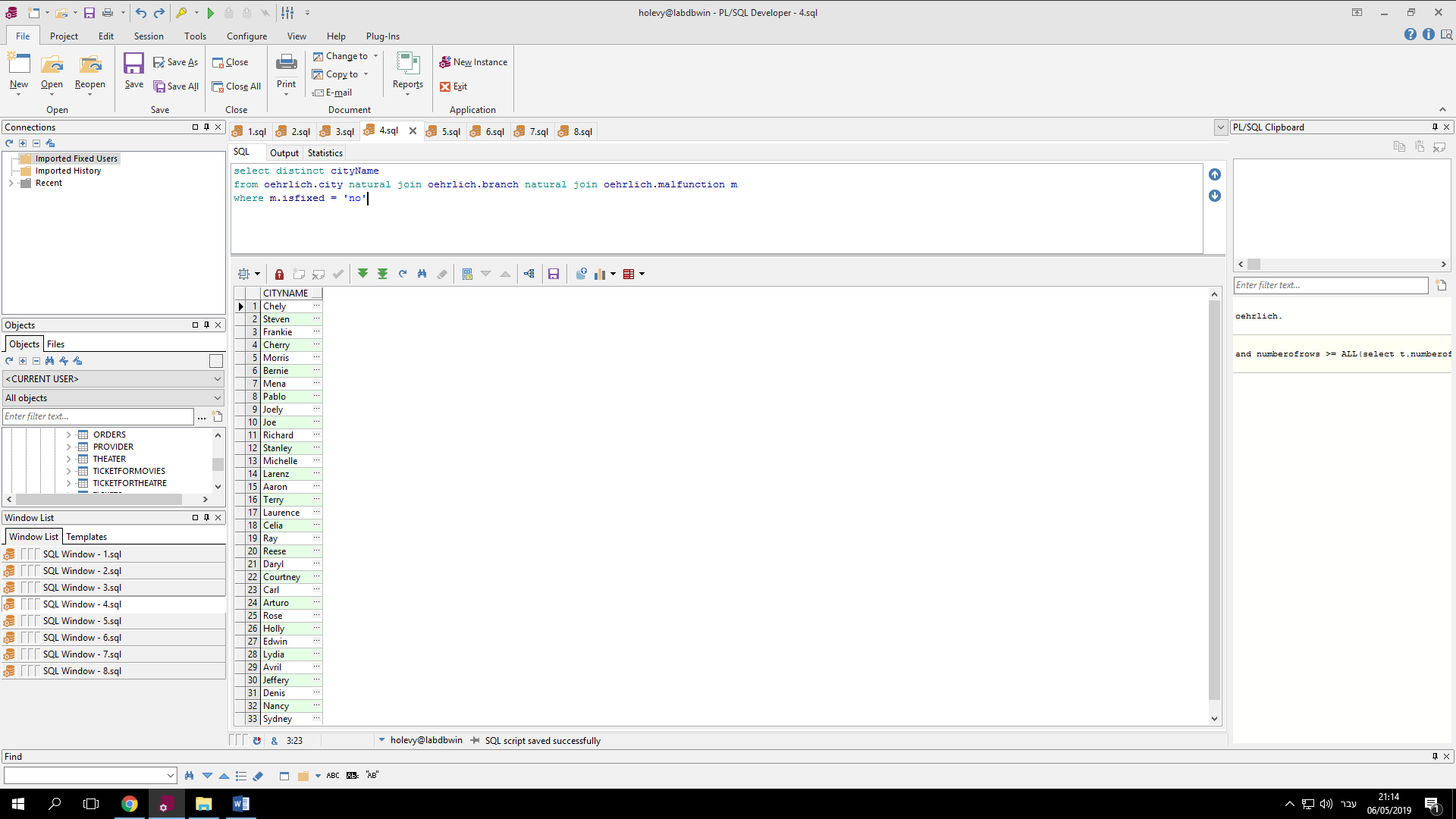


* 2. בדקנו כמה סניפים יש בבאר שבע

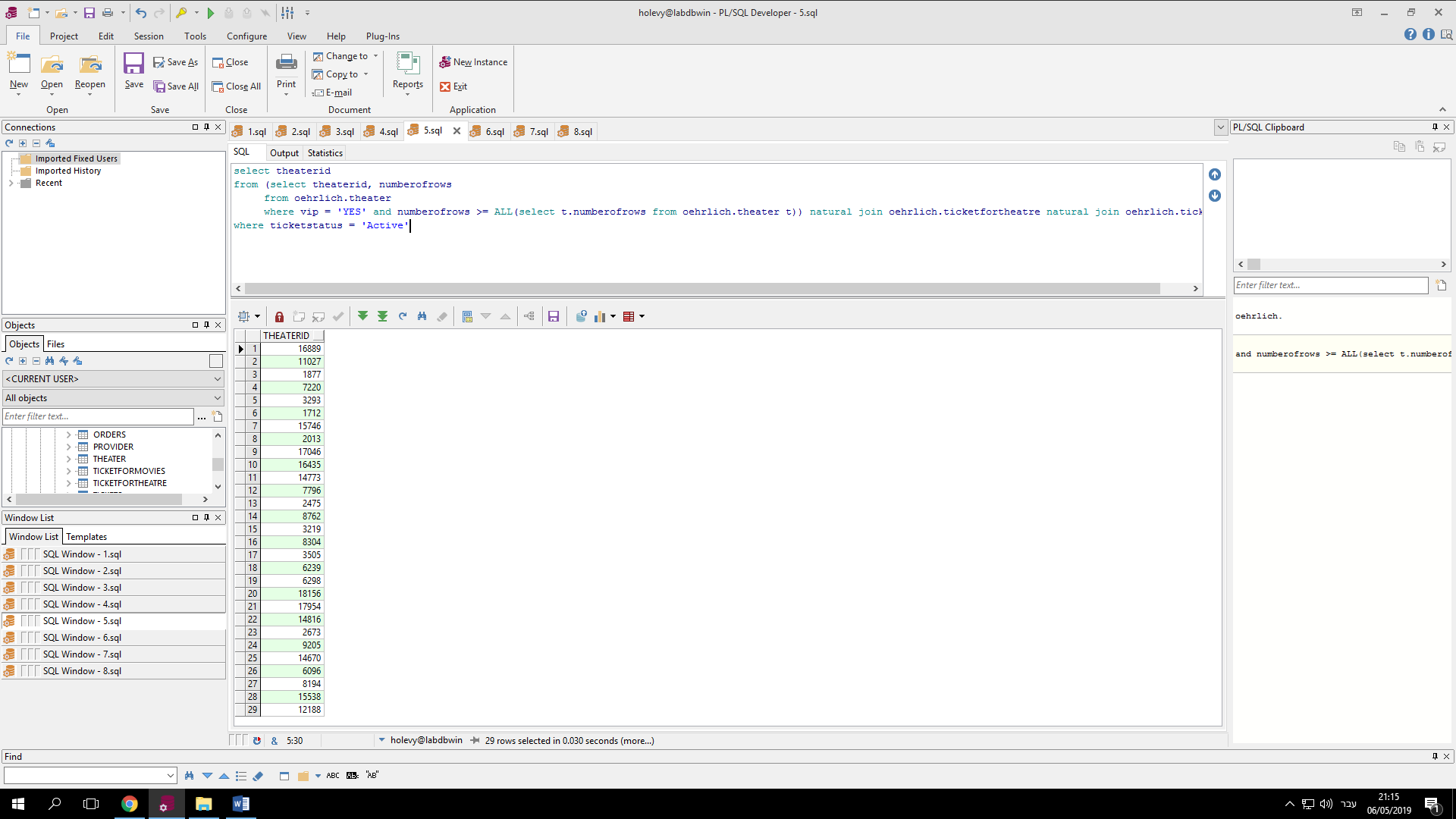


3. בדקנו באילו איזורים יש סניפים שיש בהם פחות מ-20 תקלות

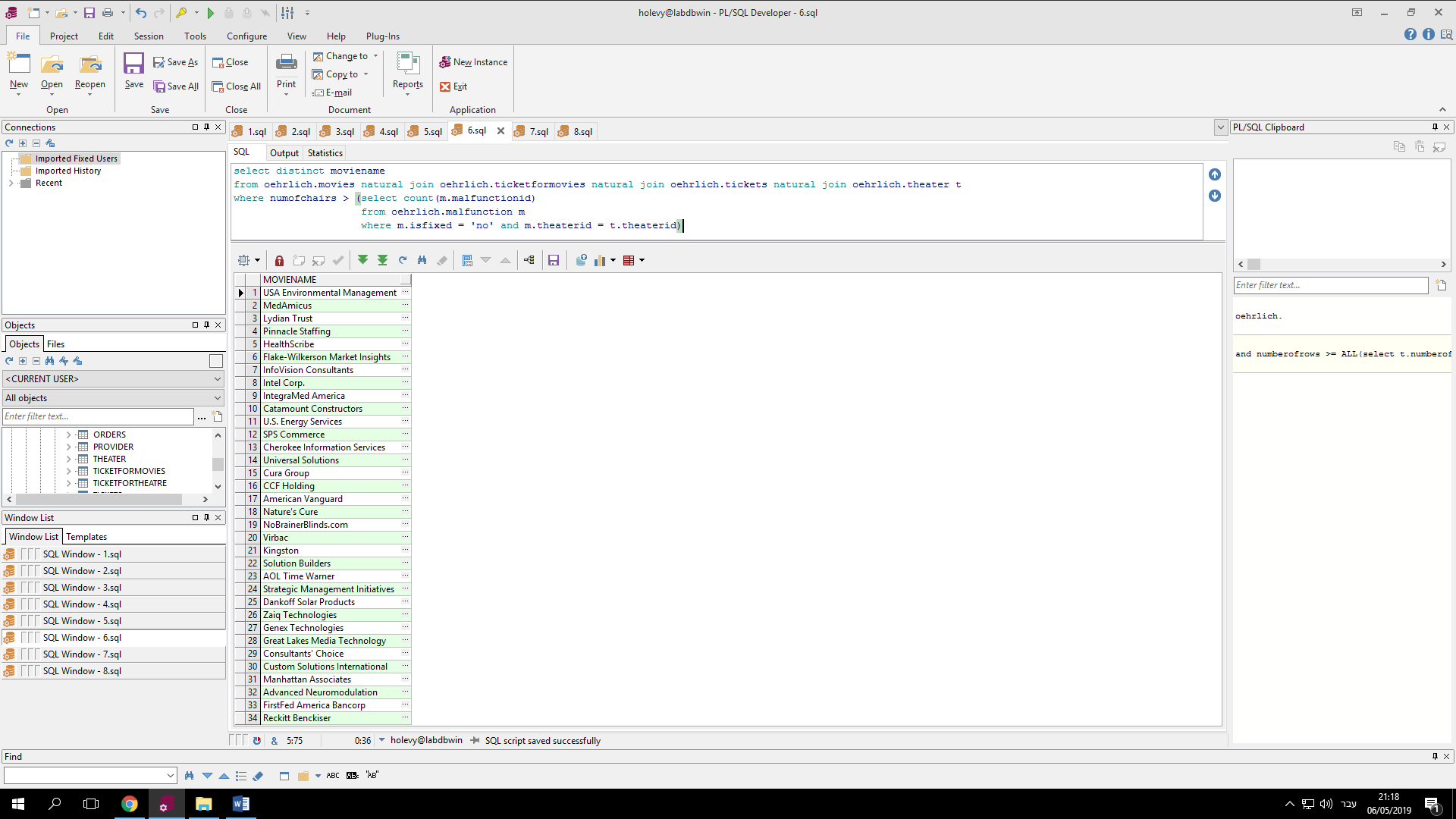


4. בדקנו באילו ערים יש סניפים שיש בהם תקלות שלא טופלו

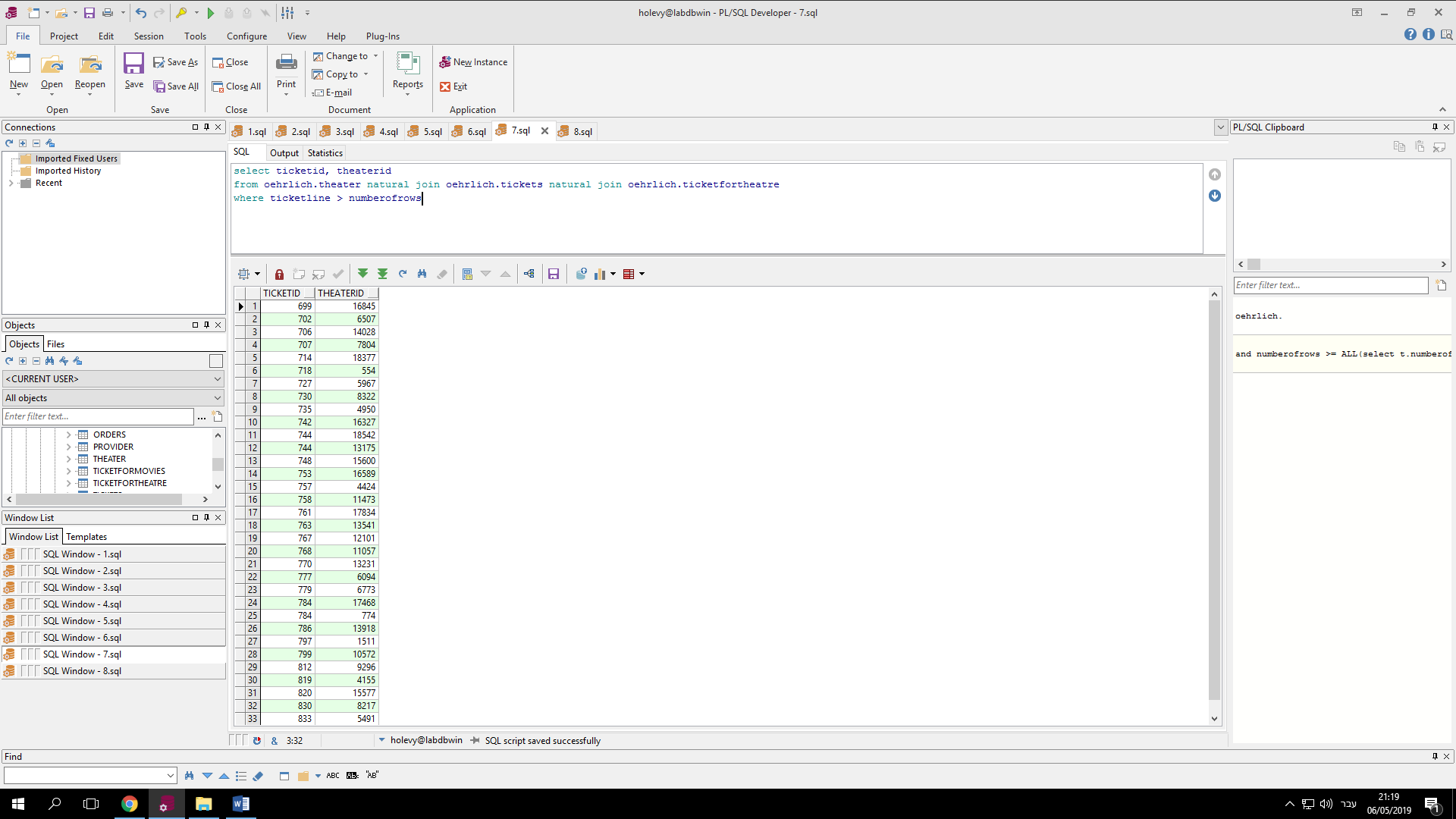
5. בדקנו באילו אולמות שהם ויאייפי ומספר השורות בהם הוא מקסימלי יש לפחות כרטיס אחד שהוא בסטטוס "פעיל"



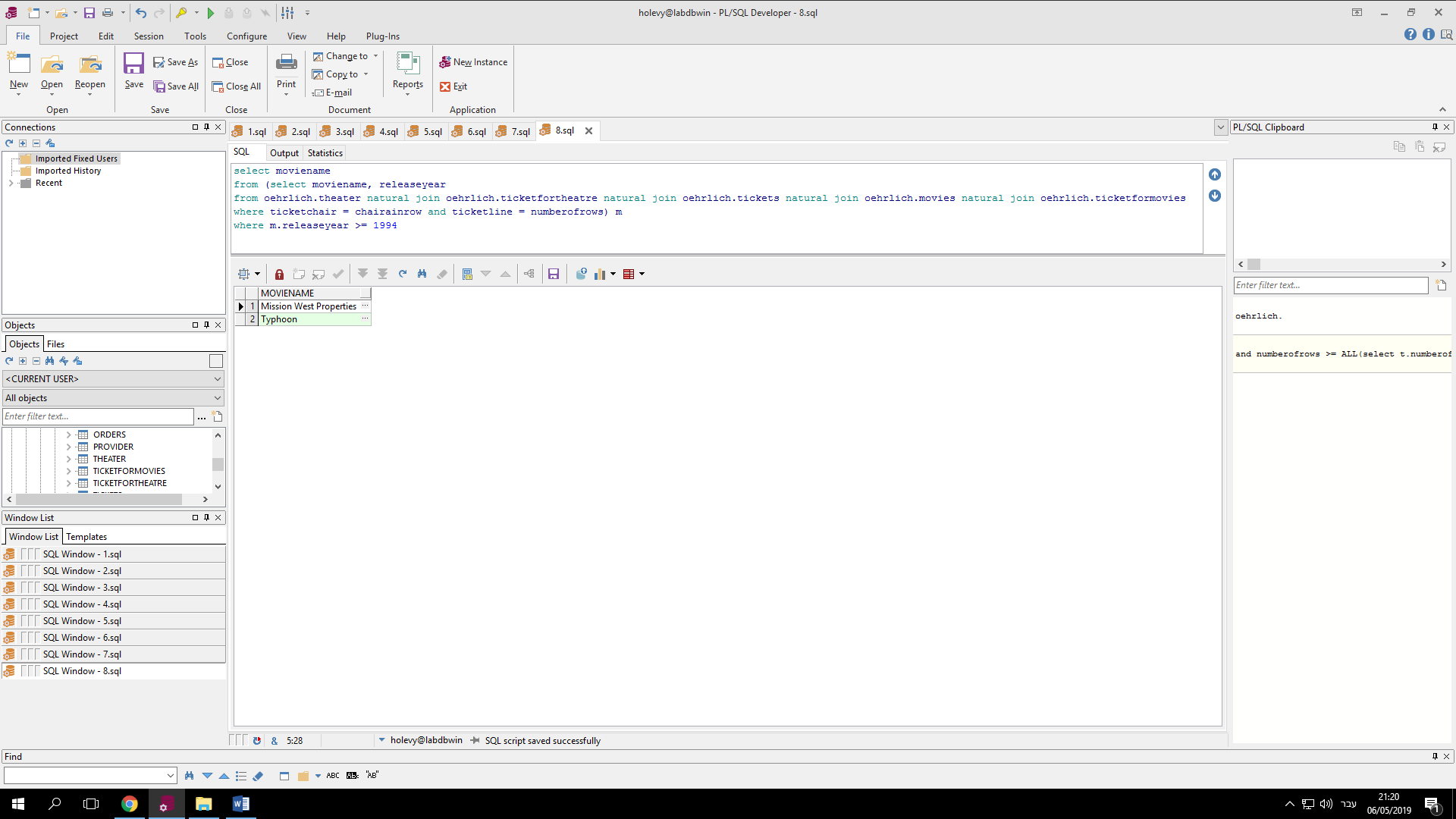
6. בדקנו אילו סרטים מוקרנים באולמות שבהם יש יותר כסאות מתקלות



7. בדקנו איזה כרטיס הזמין מקום באולם מעבר למספר השורות הקיימות באותו אולם



8. בדקנו איזה סרטים שבהם אדם הזמין כרטיס לכסא בקצה האולם יצאו משנת 1994 ואילך



## 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.

במידה ולא נגדיר אינדקס לטבלה אז בכל שאילתה על הטבלה השאילתה תגרום למעבר על כל הרשומות בטבלה עד שתמצא את כל הרשומות העונות למה שחיפשנו. כשנגדיר אינדקס מתאים אז החיפוש יהיה מהיר יותר כי הפניה לבסיס הנתונים תגרום לזה שמנוע החיפוש בבסיס הנתונים יפנה קודם לאינדקס וילך לרשומות המתאימות על פי מה שרשום באינדקס.

לכן יצרנו אינדקסים שמקצרים את תהליך ביצוע השאילתות.

את כל הבדיקות עשינו ללא cache ע"י שסגרנו את התוכנה ( ולפעמים כיבינו את המחשב ), על מנת שזה לא יושפע מהcache.

האינדקסים הנ"ל:

create index IDX\_MALFUNCTION\_IS\_FIXED on oehrlich.malfunction(isfixed);

create index IDX\_BRANCH\_BRANCHID on oehrlich.branch(branchid);

create index IDX\_AREA\_AREACOUNTRY on oehrlich.area(areacountry);

1. שאילתה מספר 1: שאלנו שאילתה המחזירה טבלה של שמות המספרים המזהים של כל הסניפים שנמצאים בישראל בעקבות בקשה של הציבור בישראל לדעת אילו סניפים קולנוע של הרשת המצליחה שלנו נמצאים בארץ הקודש.

ישנם 49,275 סניפים, ומתוכם נמצאים בישראל 391. שיפרנו את זמן ההרצה של השאילתה בערך פי 3!

האינדקס שיצרנו היה על המדינה "areaCountry" ולכן כשחיפשנו בשאילתה את המספרים המזהים של הסניפים שנמצאים בישראל היה מהיר יותר לקבל את התוצאה כי ההסתברות לקבל שם של מדינה היא גבוהה יותר.

|  |  |
| --- | --- |
| לפני אינדוקס | אחרי אינדוקס |
| 0.237 | 0.086 |

1. שאילתה מספר 3: שאלנו שאילתה המחזירה טבלה של כל שמות האיזורים שיש בהם לפחות סניף אחד שיש בו מעל 20 תקלות, שהרי ידוע שסניף עם מעל ל-20 תקלות מקבל ציון נכשל במדד הבטיחות של מכון התקנים של האו"ם ונידון לסגירה ולקנס גבוה מאוד.

ישנם 6,808 איזורים, ומתוכם 30 עונים על השאילתה. שיפרנו את זמן הריצה בערך פי 2!

האינדקס שיצרנו היה על המספר המזהה של הסניפים, לכן מבחינה הסתברותית יש לנו פחות רשומות שחוזרות על עצמן מאשר מזהה שהוא רציף ושונה מרשומה אחת לשנייה ולכן זמן החיפוש היה מהיר יותר.

|  |  |
| --- | --- |
| לפני אינדוקס | אחרי אינדוקס |
| 0.197 | 0.096 |

1. שאילתה מספר 6: שאלנו שאילתה המחזירה טבלה של כל שמות הסרטים שמוקרנים באולומת שבהם יש יותר כיסאות מתקלות, כי כך אם נניח שכל התקלות הן בכיסאות נוכל לשער שלא בכל הכיסאות יש תקלות וכך נוכל לאשר את הקרנת הסרט.

ישנם 7,195 סרטים, ומתוכם 697 עונים על השאילה. שיפרנו את זמן הריצה בזמן זעיר מאוד שלא ניתן לראות השפעה אמיתית.

האינדקס שיצרנו היה על העמודה is\_fixed ולכן מבחינה הסתברותית יש לנו פחות רשומות זהות מאשר עמודות אחרות בטבלה ולכן זמן החיפוש היה מעט קצר יותר.

|  |  |
| --- | --- |
| לפני אינדוקס | אחרי אינדוקס |
| 34.372 | 34.223 |

## 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.

1. יצרנו view עם עמודות branchid, ticketid, theatreid כיוון שרבות מהשאילתות שלנו מתייחסות לעמודות אלו ולכן יעיל יותר לעבוד עם שלושת עמודות אלו באופן נפרד, בלי התייחסות לטבלאות המלאות.

CREATE or replace view Theatre\_Branch\_Ticket as

select theaterID, branchID, ticketID

from oehrlich.malfunction natural join oehrlich.ticketfortheatre

דוגמה לשימוש בVIEW בשאילתה מספר 11

1. יצרנו view עם עמודות movieid, ticketid, theatreid כיוון שרבות מהשאילתות שלנו מתייחסות לעמודות אלו ולכן יעיל יותר לעבוד עם שלושת עמודות אלו באופן נפרד, בלי התייחסות לטבלאות המלאות.

CREATE or replace view Movie\_Ticket\_Theatre as

select MovieName, TheaterID, TicketID

from oehrlich.TicketForMovies natural join Theatre\_Branch\_Ticket natural join oehrlich.movies

1. יצרנו view עם עמודות areaName, areaId, branchId כיוון שחלק מהשאילתות שלנו מתייחסות לעמודות אלו ולכן יעיל יותר לעבוד עם שלושת עמודות אלו באופן נפרד, בלי התייחסות לטבלאות המלאות.

CREATE or replace view Area\_Branch as

select areaName, areaId, branchId

from oehrlich.area natural join oehrlich.city natural join oehrlich.branch

1. יצרנו view עם עמודות theatreID, VIP כיוון שחלק מהשאילתות שלנו מתייחסות לעמודות אלו ולכן יעיל יותר לעבוד עם שתי עמודות אלו באופן נפרד, בלי התייחסות לטבלאות המלאות.

CREATE or replace view Ticket\_VIP as

select ticketID, VIP

from oehrlich.theater natural join oehrlich.ticketfortheatre

## פונקציות

פונקציה בשפת SQL  היא צורה מיוחדת של פקודה אשר מבצעת פעולות שונות על הנתונים בבסיס הנתונים ומחזירה ערך.

פונקציה מספר 1:

בהינתן שם של עיר, הפונקציה מחזירה מספר סניפי קולנוע שיש באותה עיר

create or replace function BranchesInCity(city\_name in string) return number is

FunctionResult number;

begin

select count(branchID) into FunctionResult

from oehrlich.city c natural join oehrlich.branch

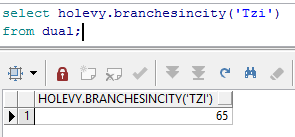
where c.cityname = city\_name

group by c.cityname;

return(FunctionResult);

end BranchesInCity;

דוגמת הרצה:



פונקציה מספר 2:

בהינתן מספר מזהה של אולם, הפונקציה מחזירה את ממוצע הרייטינג של הסרטים שהוקרנו באותו אולם

create or replace function AvgRatingInTheatre(theatre\_id in number) return float is

FunctionResult float;

begin

select AVG(rating) into FunctionResult

from oehrlich.movies natural join oehrlich.ticketformovies natural join oehrlich.tickets natural join oehrlich.ticketfortheatre natural join oehrlich.theater t

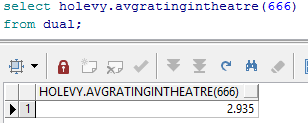
where theaterid = theatre\_id

group by theaterid;

return(FunctionResult);

end AvgRatingInTheatre;

דוגמת הרצה:



## פרוצדורות

פרוצדורה בשפת SQL  היא צורה מיוחדת של פקודה אשר מבצעת פעולות שונות על הנתונים בבסיס הנתונים.

פרוצדורה מספר 1:

הפרוצדורה מעדכנת כל כרטיס שמספר השורה שלו גדול יותר ממספר השורות המקסימלי למספר השורות המקסימלי

create or replace procedure UpdateWrongRow is

cursor t\_tickets is

select \*

from tickets

for update;

rec\_tickets tickets%rowtype;

update\_line tickets.ticketline%type;

begin

open t\_tickets;

loop

fetch t\_tickets into rec\_tickets;

exit when t\_tickets%notfound;

update\_line := 0;

if(rec\_tickets.ticketline > 25)then

update tickets t

set t.ticketline = 25

where current of t\_tickets;

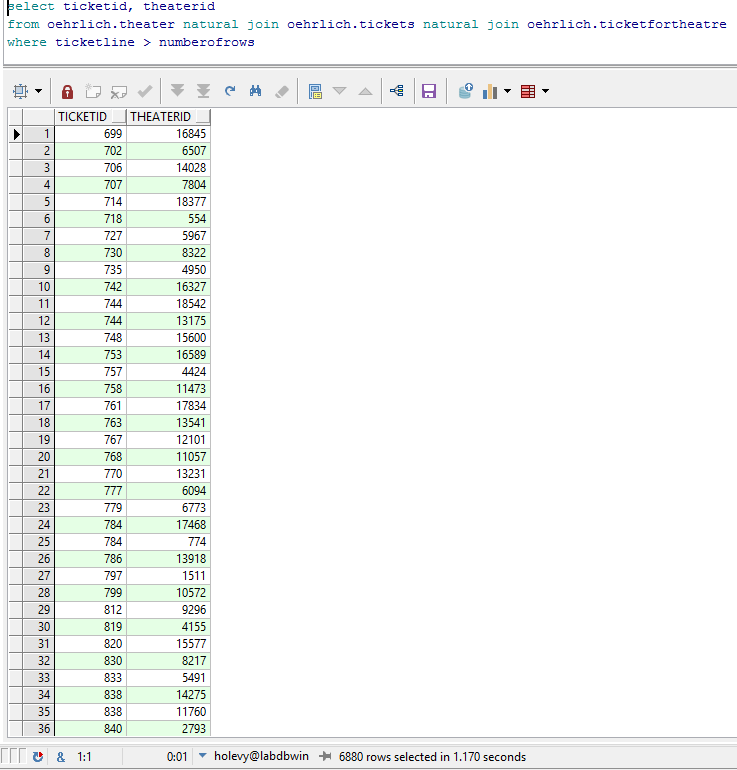
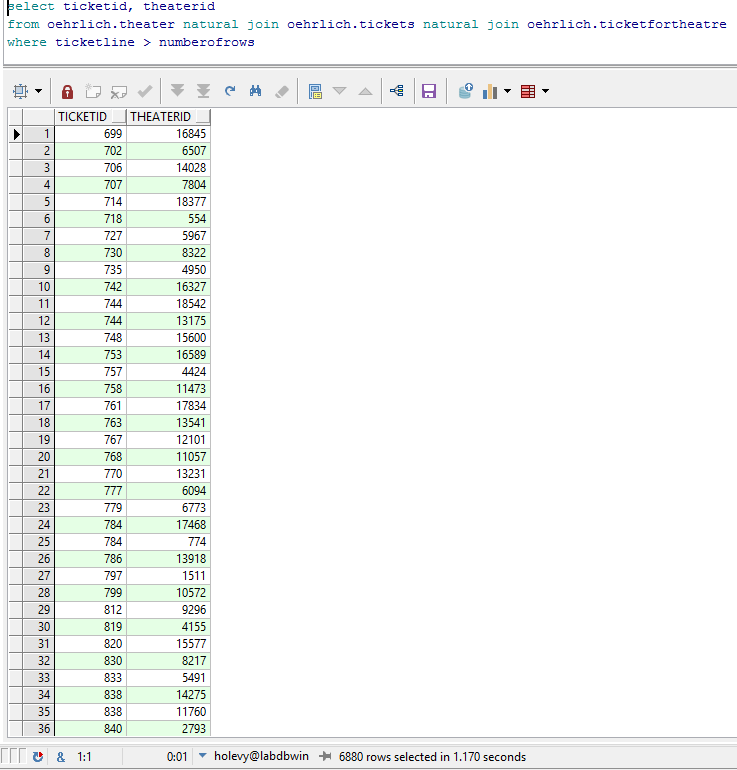
end if;

end loop;

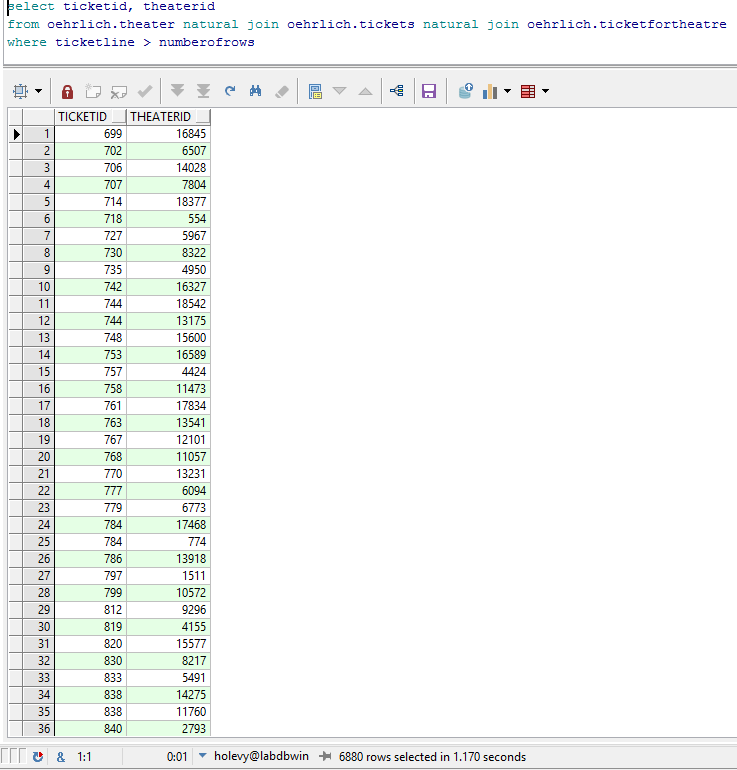
close t\_tickets;

end UpdateWrongRow;

תוצאת ההרצה של שאילתה 7 (הבודקת מקרים של חריגה זו) לפני ביצוע הפרוצדורה:



תוצאת ההרצה של שאילתה 7 (הבודקת מקרים של חריגה זו) לאחר ביצוע הפרוצדורה:



פרוצדורה מספר 2:

בהינתן שם של סרט, הפרוצדורה מכניסה למשתנה ticket\_count את מספר הכרטיסים שנמכרו לסרט הנתון.

create or replace procedure MovieData(movie\_name in string, ticket\_count out number) is

begin

select count(ticketID) into ticket\_count

from oehrlich.movies m natural join oehrlich.ticketformovies natural join oehrlich.tickets

where m.movieName = movie\_name;

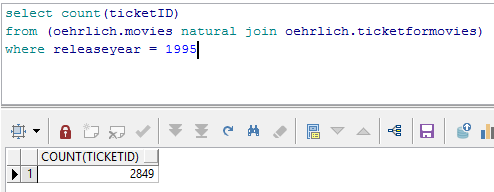
end MovieData;

## שאילתות SQL נוספות

קיבלנו תוספת של ישויות שאיתן עלינו לעבוד (סרטים, אולמות וכרטיסים), וכעת נעשה שאילתות SQL נוספות שיכללו ישויות אלו.

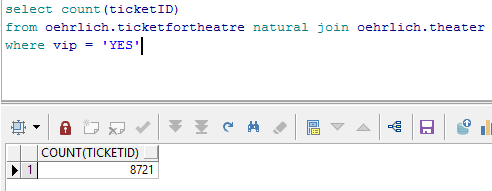
שאילתה מספר 9:

בדקנו כמה כרטיסים נמכרו בשנת 1995



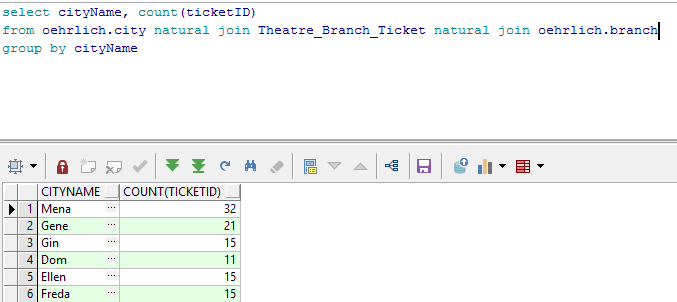
שאילתה מספר 10:

בדקנו כמה כרטיסים נמכרו לאולם VIP



שאילתה מספר 11:

בדקנו כמה כרטיסים נמכרו בכל עיר



שימוש בVIEW שיצרנו (למעלה) לעומת אי שימוש בו (למטה)