**בס"ד**

**מיני פרויקט בבסיסי נתונים**

תוכן עניינים

[מבוא 3](#_Toc517367483)

[עבודת הכנה והכרת התוכנה 4](#_Toc517367484)

[תרשים ERD 4](#_Toc517367485)

[תיאור הישויות והקשרים 5](#_Toc517367486)

[ישויות 5](#_Toc517367487)

[קשרים 5](#_Toc517367488)

[נרמול הטבלאות 5](#_Toc517367489)

[פרוקים 5](#_Toc517367490)

[תרשים DSD 6](#_Toc517367491)

[הפרויקט שלנו 7](#_Toc517367492)

[תרשים ERD 7](#_Toc517367493)

[תיאור הישויות והקשרים 7](#_Toc517367494)

[ישויות 7](#_Toc517367495)

[קשרים 8](#_Toc517367496)

[נרמול הטבלאות 8](#_Toc517367497)

[תרשים DSD 9](#_Toc517367498)

[יצירת הטבלאות 9](#_Toc517367499)

[הכנסת נתונים 10](#_Toc517367500)

[שאילתות SQL 11](#_Toc517367501)

[בחירה - SELECT 11](#_Toc517367502)

[עדכון – UPDATE 12](#_Toc517367503)

[מחיקה – DELETE 14](#_Toc517367504)

[הכנסה – INSERT 14](#_Toc517367505)

[אינדקסים 15](#_Toc517367506)

[הרשאות 18](#_Toc517367507)

[Views 19](#_Toc517367508)

[פונקציות 20](#_Toc517367509)

[20](#_Toc517367510)

[נספחים 21](#_Toc517367511)

[נספח ראשון: שאילתות ואינדקסים 21](#_Toc517367512)

[נספח שני: עדכון,מחיקה 24](#_Toc517367513)

[נספח שלישי: views 25](#_Toc517367514)

[נספח רביעי: פונקציות 25](#_Toc517367515)

# מבוא

* **מחלקת לוגיסטיקה**

מלונות: מזהה, כתובת, פרטים.

חדרים: מזהה, מלון, מחיר.

תחזוקה: צוות, מזהה מקום, סטטוס, תאריך.

* **מנהלה**

עבודות: שם, תיאור.

עובדים: מזהה, שם, שכר, כתובת, שיוך לקבוצה.

קבוצות: מזהה, שם, תיאור, תקציב, מזהה של מלון.

חשבונות: מזהה טרנזקציה, תיאור, סכום, תאריך.

בקשה: קבוצה, העברה.

* **מחלקת לקוחות**

לקוח: מזהים, פרטים.

הזמנות: מספר חבר, תאריך.

היסטוריה: מזהה, דירוג.

תלונות: מספר תלונה, סוג, תיאור, חומרה, מספר לקוח.

* **מחלקת אספקה**

ספקים: שם, מזהה, קטגוריה, איזור.

פריטים: תיאור, מזהה.

תשלומים: מזהה, פרטי דרכי תשלום.

* **מחלקת מתקנים**

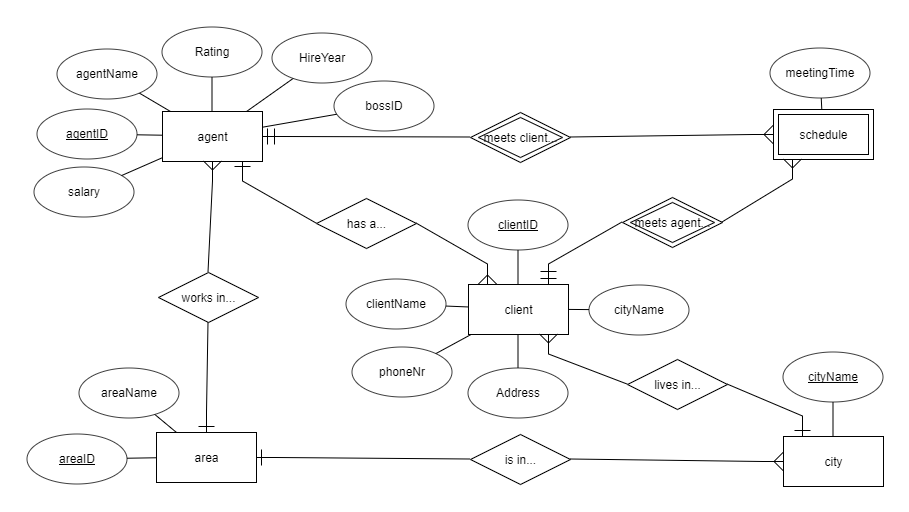
מתקנים: מזהה מתקן, שם מתקן, תפוסה, שעות פעילות, מזהה קבוצת עובדים, מזהה מלון.

חשבונות: מזהה לקוח, סכום החשבון, מזהה מתקן, מזהה חשבון.

בקשות אספקה: מזהה מתקן, מזהה מוצר, מזהה בקשת אספקה, תאריך בקשת אספקה.

# עבודת הכנה והכרת התוכנה

## תרשים ERD



## תיאור הישויות והקשרים

### ישויות

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

### קשרים

* לכל סוכן- יש אזור עבודה יחידי, יכולים להיות הרבה לקוחות ויכולים להיות לו הרבה פגישות עם לקוחות.
* לכל אזור- יכולות להיות כמה ערים ויכולים להיות הרבה סוכנים שעובדים באזור זה.
* לכל עיר- יכולים להיות הרבה לקוחות שגרים בה והיא יכולה להיות באזור יחידי.
* 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

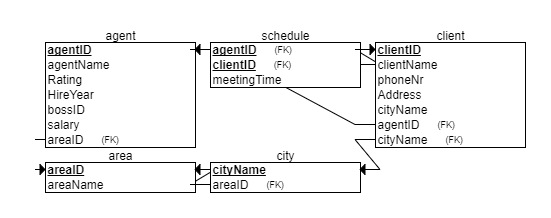
* Schedule (agentID, clientID, meetingTime)
* Agent (agentID, rating, hireYear, boosID, salary, areaID)
* Client (clientID, clientName, address, phoneNr, cityName)
* CityName (cityName, areaID)
* Area (areaID, areaName)

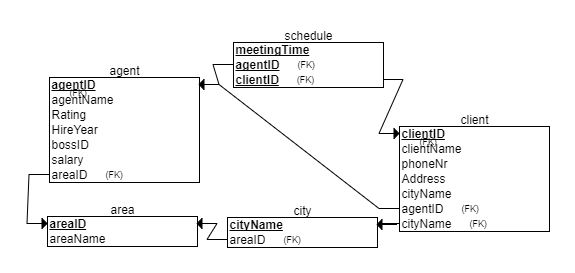
### פרוקים

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.

אך זה לא כמו מה שמומש ב- create table שניתן לנו כי שם ב- schedule מופיע גם ה- meetingTime כ- praimery key ופה הוא לא מופיע.

לכן נוסיף זאת לתרשים שלנו באופן ידני, ונקבל את התרשים הבא.

# הפרויקט שלנו

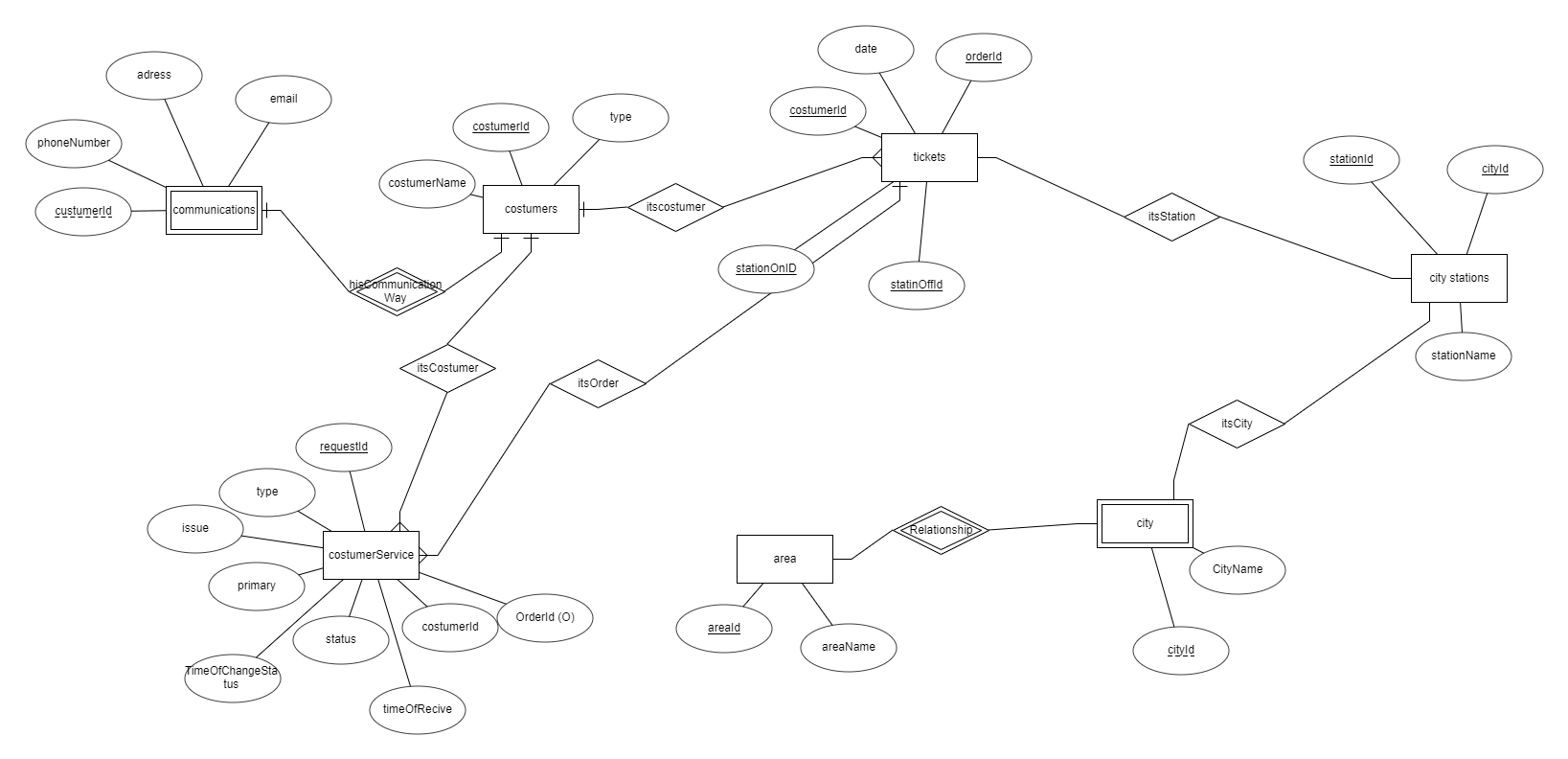
*המחלקה שלנו היא מחלקת שירות לקוחות, המחלקה מקבלת פניות (מידע, שיבוח או תלונות) ומטפלת בהם. נשתמש בבסיס הנתונים לצורך קבלת מידע סטטיסטי לצורך שיפור ויעול השירות.*

*את כל הקבצים המובאים להלן אפשר למצוא בתוך הגיט בכתובת:*

<https://github.com/Yechel/DataBaseProject5779>

## תרשים ERD

כפי שהזכרנו, במחלקה שלנו, ישנן 7 ישויות: שירות לקוחות, לקוח, דרכי תקשורת, כרטיס נסיעה, תחנות נסיעה, ערים ואיזור. בשלב הראשון יצרנו תרשים ERD שיתאר את הקשרים בין הישויות הללו ואת התכונות שלהן.

**את פורמט הerdplus שלו אפשר למצוא [בגיט](https://github.com/Yechel/DataBaseProject5779).

## תיאור הישויות והקשרים

### ישויות

* costumers– ישות זאת מייצגת את הלקוח אשר מבצע נסיעה (קנה כרטיס) או שהינו רשום במערכת וברצונו לקבל מידע אודות פעילות הרכבות. ישות זו היא חזה מכיוון שאינה תלויה בישיות אחרות.
* costumerId– מספר מזהה של הלקוח (PK)
* costumerName– שם הלקוח (שם ושם משפחה)
* type – סוג הלקוח, ('רגיל', 'מתחת ל18' , 'מעל '70, 'סטודנט', 'חייל' וכו')
* communications – ישות זאת מייצגת את דרכי ההתקשרות עם הלקוח.

ישות זאת הינה חלשה, כיוון שהיא תלויה בישות הלקוח.

* costumerId – מספר מזהה של הלקוח (PK)(FK)
* email – כתובת הדוא"ל של הלקוח
* phone– מספר הטלפון של הלקוח
* adress – כתובת הלקוח
* tickets – ישות זאת מייצגת את הכרטיסים שנרכשו.

ישות זאת הינה חלשה, כיוון שהיא תלויה בין היתר בלקוח, ובתחנות העליה והירידה.

* orderId – מספר מזהה של ההזמנה (הכרטיס) (PK)
* Date – תאריך הרכישה
* costumerId – מספר הלקוח (FK)
* stationOnId – מספר תחנת עליה (FK)
* statinOffId – מספר תחנת ירידה (FK)
* city stations– ישות זאת מייצגת את התחנות של הרכבת.

ישות זאת הינה חזקה, כיוון שהיא לא תלויה בישות אחרת לצורך ההגדרה היחודית שלה.

* stationId– מספר המזהה של התחנה (PK)
* cityId – מספר המזהה השל העיר (FK)
* stationName – שם התחנה
* city – ישות זאת מייצגת את הערים בהן נמצאות תחנות הרכבת.

ישות זאת הינה חלשה, כיוון שהיא תלויה בישות האיזור.

* cityId – מספר מזהה של העיר(PK)
* areaId – מספר המזהה של האיזור בו נמצאת העיר (FK)
* CityName – שם העיר.
* area – ישות זאת מייצגת את האזורים שבהם נמצאות ערים עם תחנות רכבת.

ישות זאת הינה חזקה, כיוון שאינה תלויה בשם יושות אחרת.

* areaId – המספק המזהה של האיזור (PK)
* areaName – שם האיזור
* costumerService– ישות זאת מייצגת את תיעוד שירות הלקוחות.

ישות זאת הינה חזקה, כיוון שהגדרתה אינה תלויה בשאר ישויות.

* requestId– מספר הפניה (PK)
* Type – סוג הפניה (תלונה, שיבוח, או מידע)
* Issue – תוכן הפניה
* Primary – תעדוף הפניות (ממוספר מ1 החשוב ביותר עד 5 – הכי פחות חשוב)
* Status – מצב הפניה (חדש, ממתין לטיפול, בטיפול, הסתיים, התבטל)
* TimeOfChangeStatus – זמן שינוי הסטטוס האחרון
* timeOfRecive – זמן קבלת הפניה
* costumerId – מספר הלקוח
* OrderId – מספר ההזמנה (אופציונאלי משום שיכול להיות שלקוח פנה למידע בלבד)

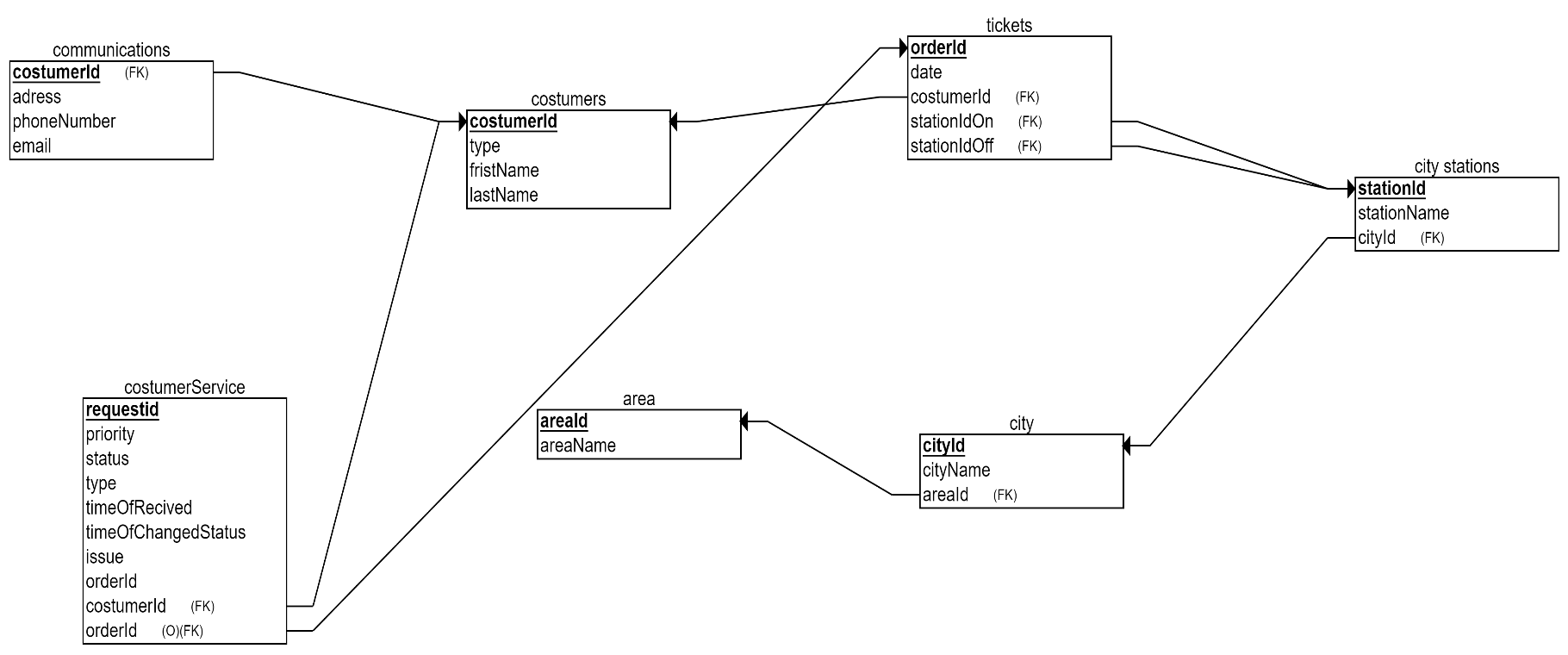
### קשרים

* hisCommunicationWay– הקשר בין costumers לבין communication. הקשר הינו חלש, כיוון שמקשר בין ישות חלשה לחזקה. הקשר הוא 1:1 משום שדרכי ההתקשרות מהווים טבל המשך לטבלת הלקוח.
* itscostumer– הקשר בין costumers לבין costumerService. הקשר הינו חזק, כיוון שמקשר בין ישות חזקה לחזקה. הקשר הוא M:1 משום שלקוח יכול לפנות הרבה פעמים לשירות הלקוחות.
* itscostumer– הקשר בין costumers לבין tickes. הקשר הינו חזק, כיוון שמקשר בין ישות חזקה לחזקה. הקשר הוא M:1 משום שללקוח יכולים להיות הרבה כרטיסים.
* itsOrder– הקשר בין tickets לבין costumerService. הקשר הינו חזק, כיוון שמקשר בין ישות חזקה לחזקה. הקשר הוא M:1 משום שיכולים להיות המון פניות על כרטיס אחד.
* itsStation– הקשר בין tickets לבין city stations. הקשר הינו חזק, כיוון שמקשר בין ישות חזקה לחזקה. הקשר הוא M:1 משום שיכולים להיות המון כרטיסים עבור תחנה אחת.
* itsCity– הקשר בין cityלבין city stations. הקשר הינו חזק, כיוון שמקשר בין ישות חזקה לחזקה. הקשר הוא M:1 משום שיכולים להיות המון תחנות בעיר אחת.
* ItsArea– הקשר בין city לבין area. הקשר הינו חלש, כיוון שמקשר בין ישות חזקה לחלשה. הקשר הוא M:1 משום שבאיזור אחד יכול להיות כמה ערים.

## 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: מחלקת מתקני המלון. את התרשים אפשק למצוא בכתובת [הגיט](https://github.com/Yechel/DataBaseProject5779)

## 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.

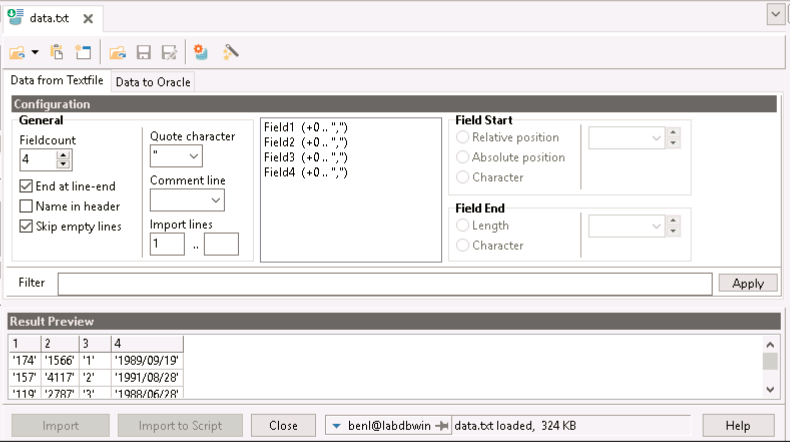
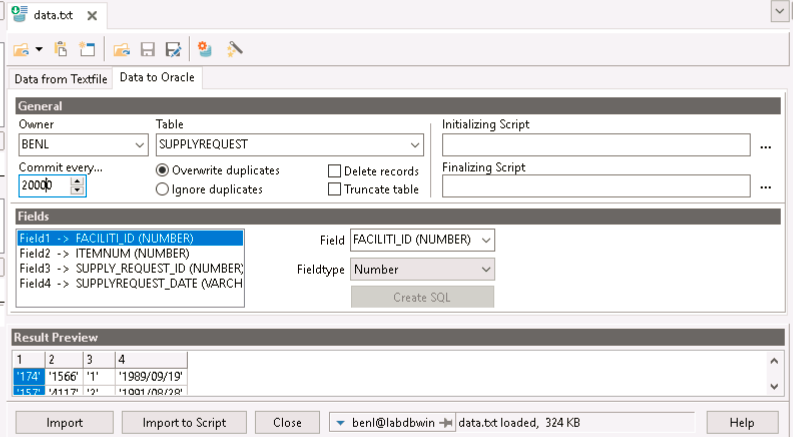
יצרנו קוד לייצור הטבלאות באמצעות export SQL של האתר erdPlus, יצרנו קובץ SQL ואז העתקנו את קוד ה- SQL של כל טבלה אל תוכנת ה-plsql לשם יצירת הטבלאות בפועל את קובץ הטקסט אפשר למצוא [בגיט](https://github.com/Yechel/DataBaseProject5779)

## הכנסת נתונים

על מנת לאכלס את הטבלאות שיצרנו בנתונים, בdata generator של plsql אותו אפשר למצוא [בגיט](https://github.com/Yechel/DataBaseProject5779)

בcostumer service שתמשנו גם ב data generator אבל כדי לוודא שזמן הטיפול הוא יהיה אחרי זמן הקבלה לקחנו את הscript שנוצר ובעזרת script של פייתון שינינו את הנתונים של זמן הטיפול. את כל הנתונים החדשים והscript של פייתון אפשר למצוא [בגיט](https://github.com/Yechel/DataBaseProject5779)

את קובץ info\_after.text הכנסנו לdata generator בצורת script.

אחר כך השתמשנו בtext importer בתוכנת ה- PL/SQL כדי להכניס את הנתונים לטבלאות. כך לדוגמא, הכנסנו נתונים לתוך הטבלה supplyrequest:

## שאילתות SQL

### בחירה - SELECT

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

הנהלת הרכבת מעוניינת להרחיב את שירותי הרכבת לאזור הפריפריה. הדבר מצריך השקעה מרובה(בניית תשתיות, פרסום מכרזים, גיוס עובדים וכיוצא בזה). לצורך כך ההנהלה מעוניינת לדעת כמה פרטים על התשתית והשירות הקיים כיום כדי לשקול את היקף ו כדאיות ההשקעה. להלן כמה שאילתות שההנהלה ביקשה לברר:

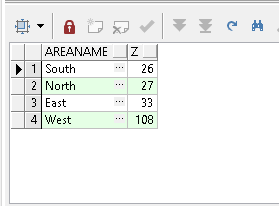
1. פריסה של כמות התחנות הקיימות כיום לפי אזורים:

select areaname, count(areaid) as z from CITY\_STATIONS

NATURAL JOIN CITY NAtural join area

GROUP BY areaname

order by z

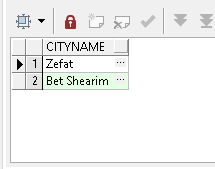


1. בכמה ערים יש לכל היותר שתי תחנות?

NATURAL JOIN CITY

GROUP BY CITYNAME

HAVING COUNT(CITYID)<=2



**3)** מהי העיר שיש בה הכי מעט תחנות?

select CITYNAME from CITY\_STATIONS T

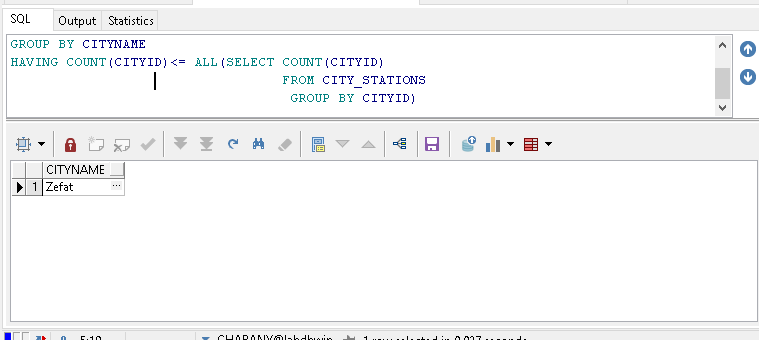
NATURAL JOIN CITY

GROUP BY CITYNAME

HAVING COUNT(CITYID)<= ALL(SELECT COUNT(CITYID)

FROM CITY\_STATIONS

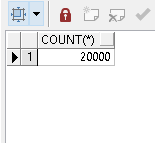
GROUP BY CITYID)



נושא נוסף ש**ההנהלה ביקשה לקבל עבורו מידע הוא "שירות לקוחות". ההנהלה מעוניינת לדעת מהו מספר הלקוחות השמורים במערכת, מהי הפנייה השכיחה ביותר של הלקוחות לצורך שימור הלקוחות והתייעלות המערכת.**

1. **מהו מספר הלקוחות ששמורים במערכת?**

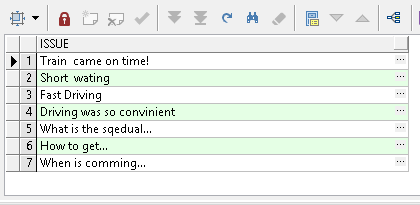
select COUNT (\*) from COMMUNICATIONS



5 **) מה הנושאים שהלקוחות בוחרים להתלונן עליו או שדווקא בוחרים לציין לשבח?**

select issue from COSTUMERSERVICE t

group by issuE

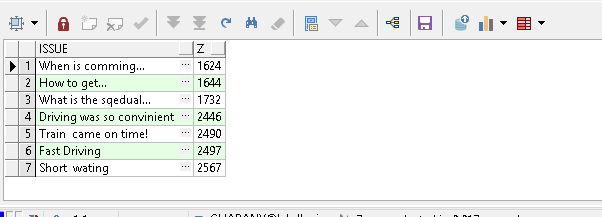


**6) כמה פניות התקבלו עבור כל נושא?**

select issue,COUNT(ISSUE) AS Z from COSTUMERSERVICE t

group by issuE

ORDER BY Z



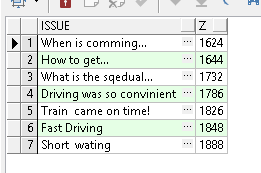
**7) כמה מתוך הפניות עדיין פתוחות (כלומר בטיפול)?**

select issue,COUNT(ISSUE) AS Z from COSTUMERSERVICE t

WHERE TYPE!='finish'

group by issuE

ORDER BY Z



הנהלת החברה ביקשה לברר באיזה עיר שבו עלו הנוסעים אחוז התלונות הוא הגבוהה ביותר

select cityId, sum(complains) as sumcomcity from (

(select s.stationIdOn as sstationid,count(\*) as complains from

((select cse.orderid as cseID from

COSTUMERSERVICE cse where cse.type = 'complain')

inner join

(select t.orderId,t.stationIdOn from tickets t) s

on cseId = s.orderId)

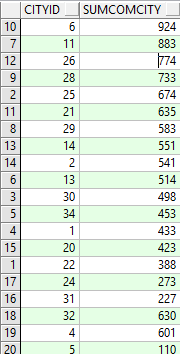
group by s.stationIdOn)

inner join

(select cityS.stationid,cityS.cityid from city\_stations cityS) a

on a.stationid = sstationid)

group by (cityid)



**שאילתת insert**

בעקבות הממצאים ההנהלה החליטה בשיתוף עם משרד התחבורה ומשרד האוצר להרחיב את השירות לתושבי הצפון. לצורך כך נבנו שתי תחנות חדשות בצפון: תחנה נוספת בעיר צפת ותחנה חדשה בעיר קצרין.

INSERT INTO CITY VALUES (38,'KAZHERIN',5)

INSERT INTO CITY\_STATIONS VALUES (201,'LEV-HAIR',38)

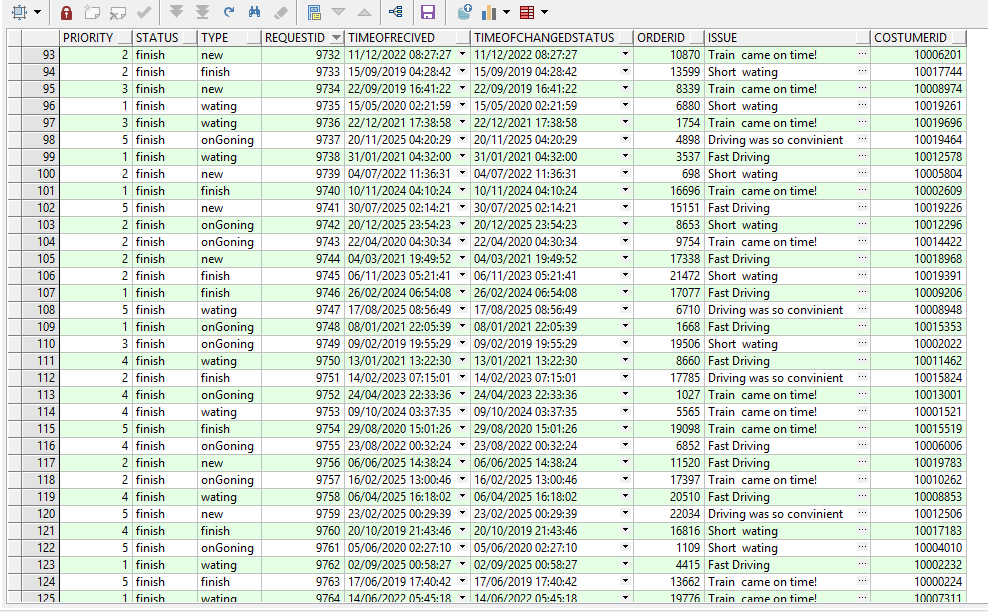
INSERT INTO CITY\_STATIONS VALUES (202,'ZIV-HOSPITAL',5)

**י**צירת view ו-update

**במעבר על בסיס הנתונים התגלה באג, שבו הוכנס בטעות סטאטוס לא נכון בעמודת הtype. המזל הוא שמבדיקה מעמיקה עלה כי רק נתונים מסוג ‘adore’ היו נגועים בבאג, מבדיקה נוספת התגלה שישנן מילות מפתח שאפשר לעקוב אחריהן ולשנות את הסטאטוס.**

update COSTUMERSERVICE t set t.type = 'adore' where ((t.issue like '%Train%came on time%' and t.issue not like '%didn%') or t.issue like '%fast driving%' or t.issue like '%so convinient%' or t.issue like '%Short wating%' or t.issue like '%Fast Driving%');

select \* from COSTUMERSERVICE t



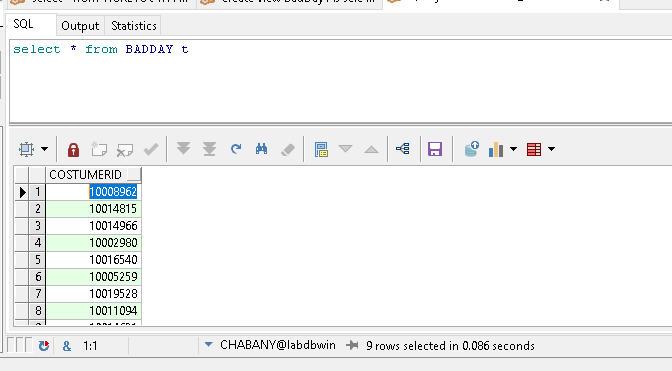
**בתאריך 06-04-2019 התרחשה שביתה כללית של כל עובדי הרכבת בחסות הסתדרות המורים – השביתה גרמה לנוסעים רבים תסכול וזעם על כך שנאלצו למצוא אמצעי תחבורה אחרת כדי להגיע ליעדם – כמחווה של רצון טוב הנהלת הרכבת החליטה לפצות את כל הנוסעים שרכשו כרטיסים ביום הזה ולשדרג את מעמדם כלקוחות אח"מ- דבר שיקנה להם מגוון הטבות .**

**לצורך יצרנו מבט כך:**

create view badDay As select costumerid from tickets

where dateorder = TO\_DATE('06/04/2019','DD-MM-YYYY')

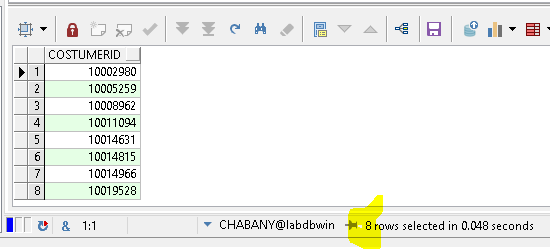
**בדיקה של כמות הנוסעים הראתה שמדובר בסך הכל בתשעה נוסעים.**



**מתוך תשעת הנוסעים – שמונה מהם לא היו בדרגה של VIP :**

select costumerid from COSTUMERS where type !='Vip' intersect

select \* from badday



**את הלקוחות הללו שיפרנו בהעלאת הדרגה**

UPDATE (select costumerid from COSTUMERS where type !='Vip' intersect

select \* from badday) SET Type ='Vip'

**Indexes**

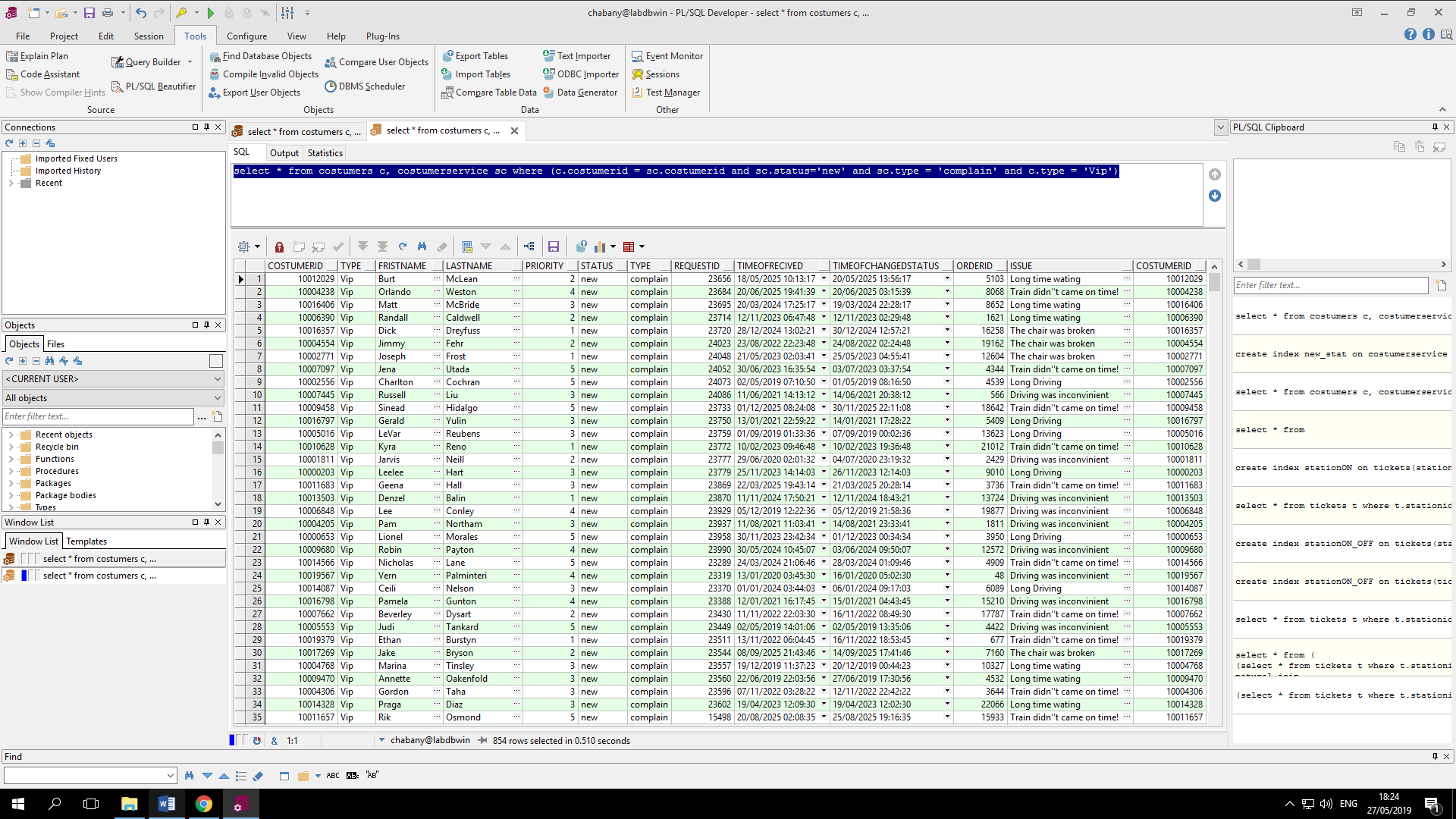
אינדקסים עוזרים למצוא במהירות גדולה יותר נתונים שנשמרו בטבלאות בבסיס הנתונים. אפשר לדמות את האינדקסים כמו מראה מקום בספר. במקום שנקרא את כל הספר כדי למצוא את מה שאנחנו מחפשים נלך למראה מקום שיראה לנו את כל המקומות שבהם מוזכר הנושא שאנחנו מחפשים. השימוש באינדקסים יחסוך לנו זמן ויהפוך את תהליך החיפוש ליעיל 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 (הקוד בנספח הראשון) שמקצרים את תהליך ביצוע השאילתות.

הצלחה:

select \* from costumers c, costumerservice sc where (c.costumerid = sc.costumerid and sc.status='new' and sc.type = 'complain' and c.type = 'Vip')



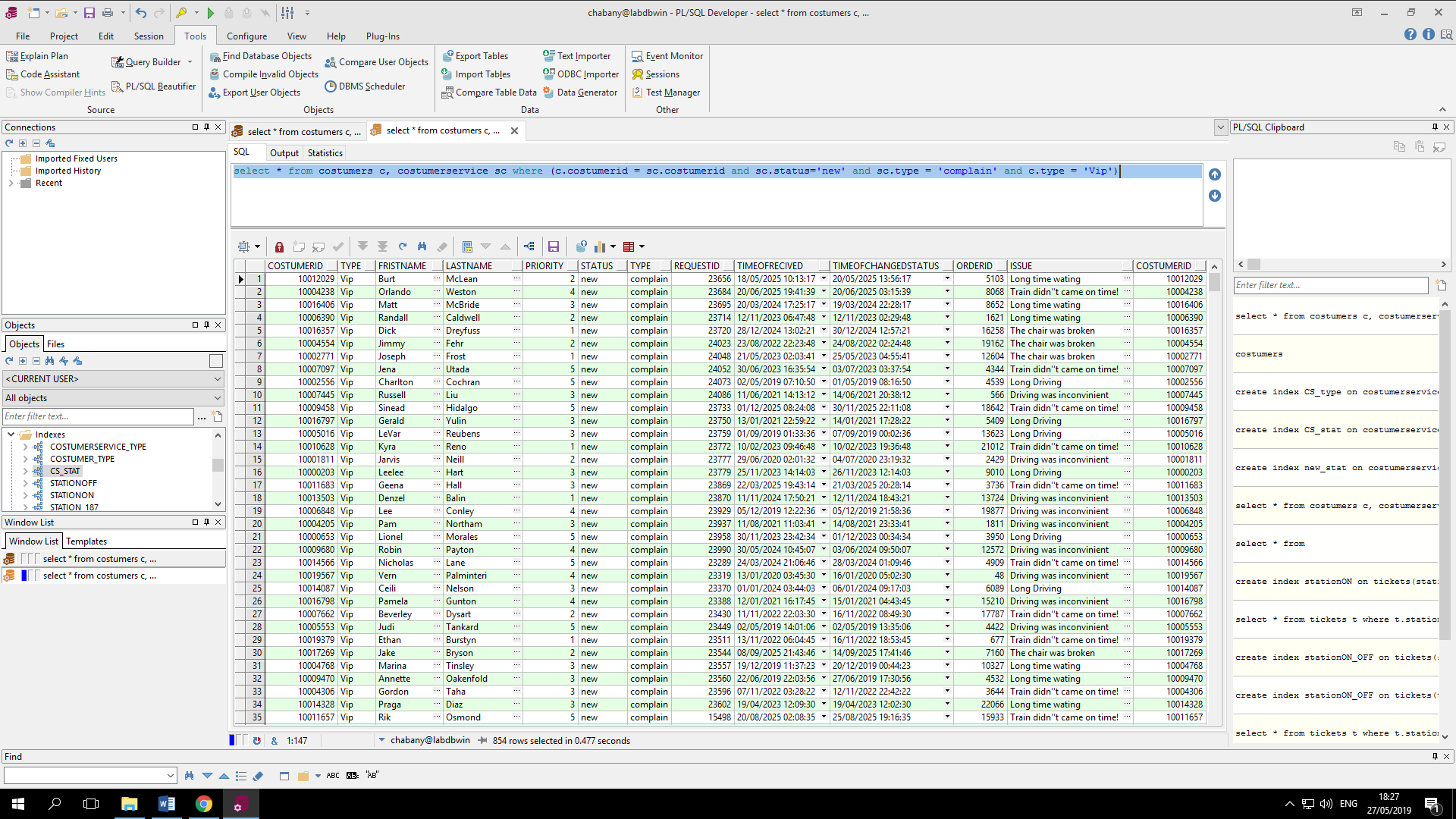
הכנסת האינדקס הבר תאפשר חיפוש רק של האינדקס "new" בעמודת הstatus

create index CS\_stat on costumerservice (status);

create index CS\_type on costumerservice (type);

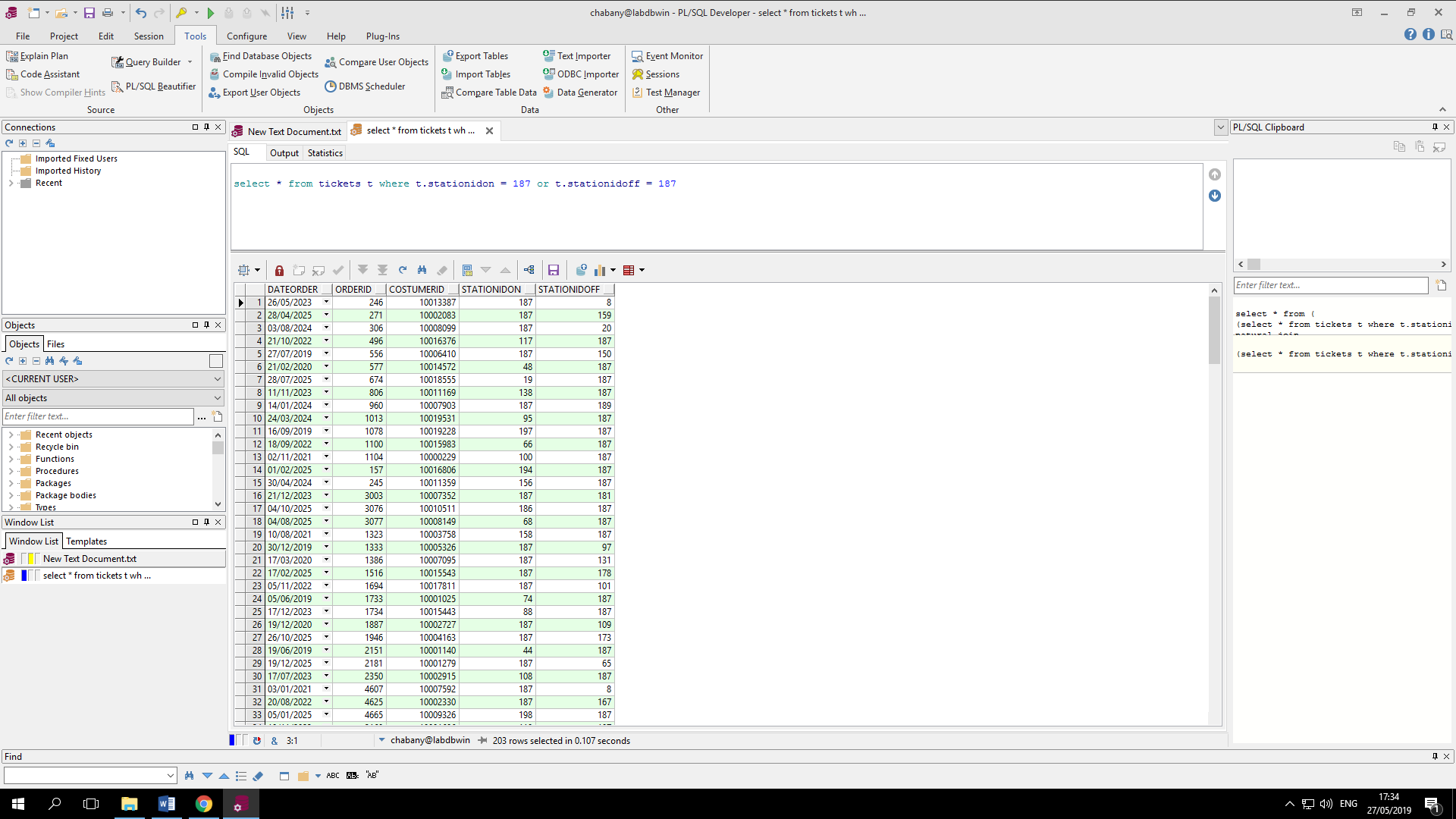
create index C\_type on costumers (type);

אכן התוצאה הייתה קצרה יותר



2.

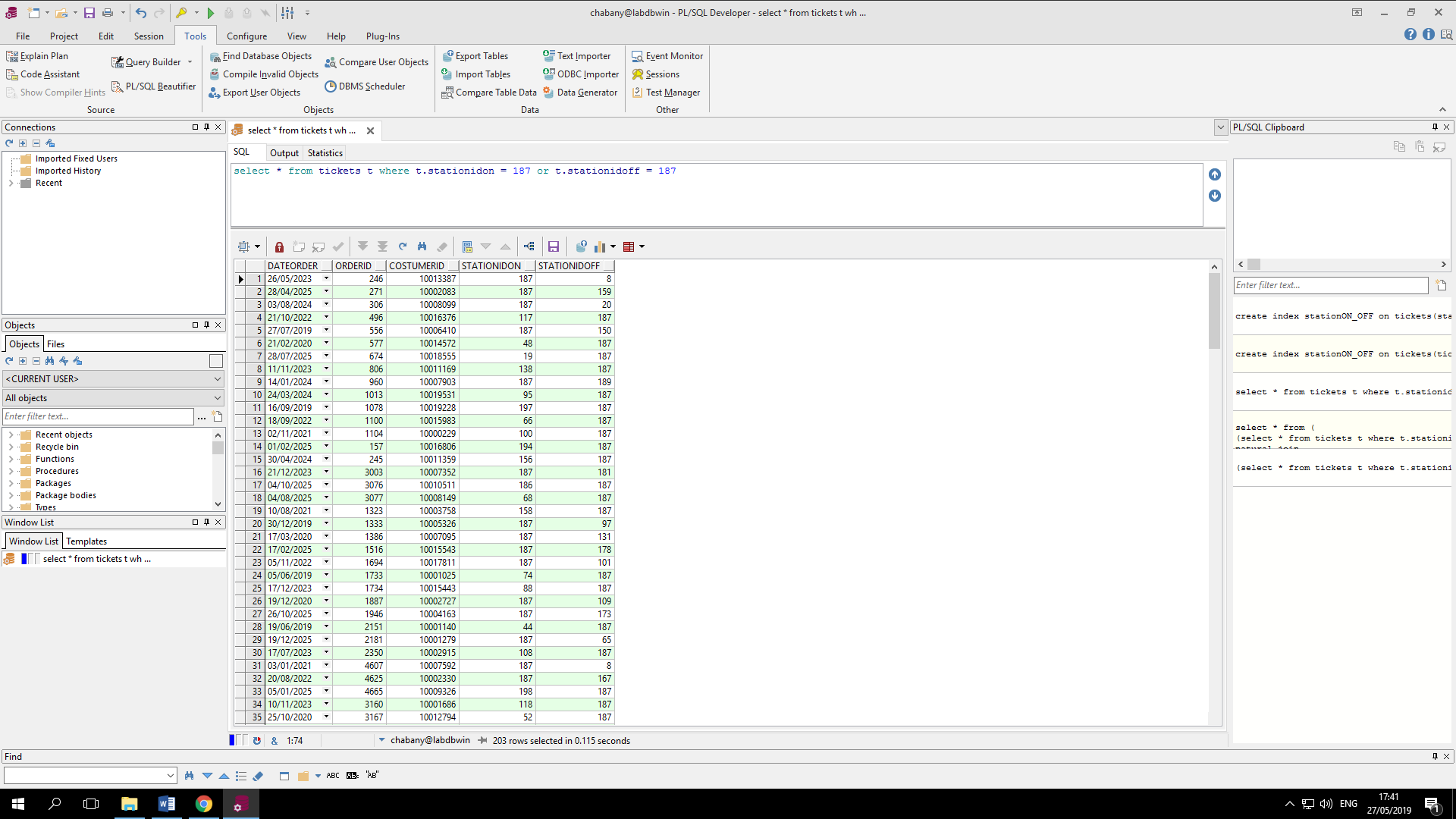
select \* from tickets t where t.stationidon = 187 or t.stationidoff = 187



הוספנו אינדקס עבור 2 עמודות:

create index stationON\_OFF on tickets(stationidon,stationidoff)

והתוצאה הייתה ללא שינוי:

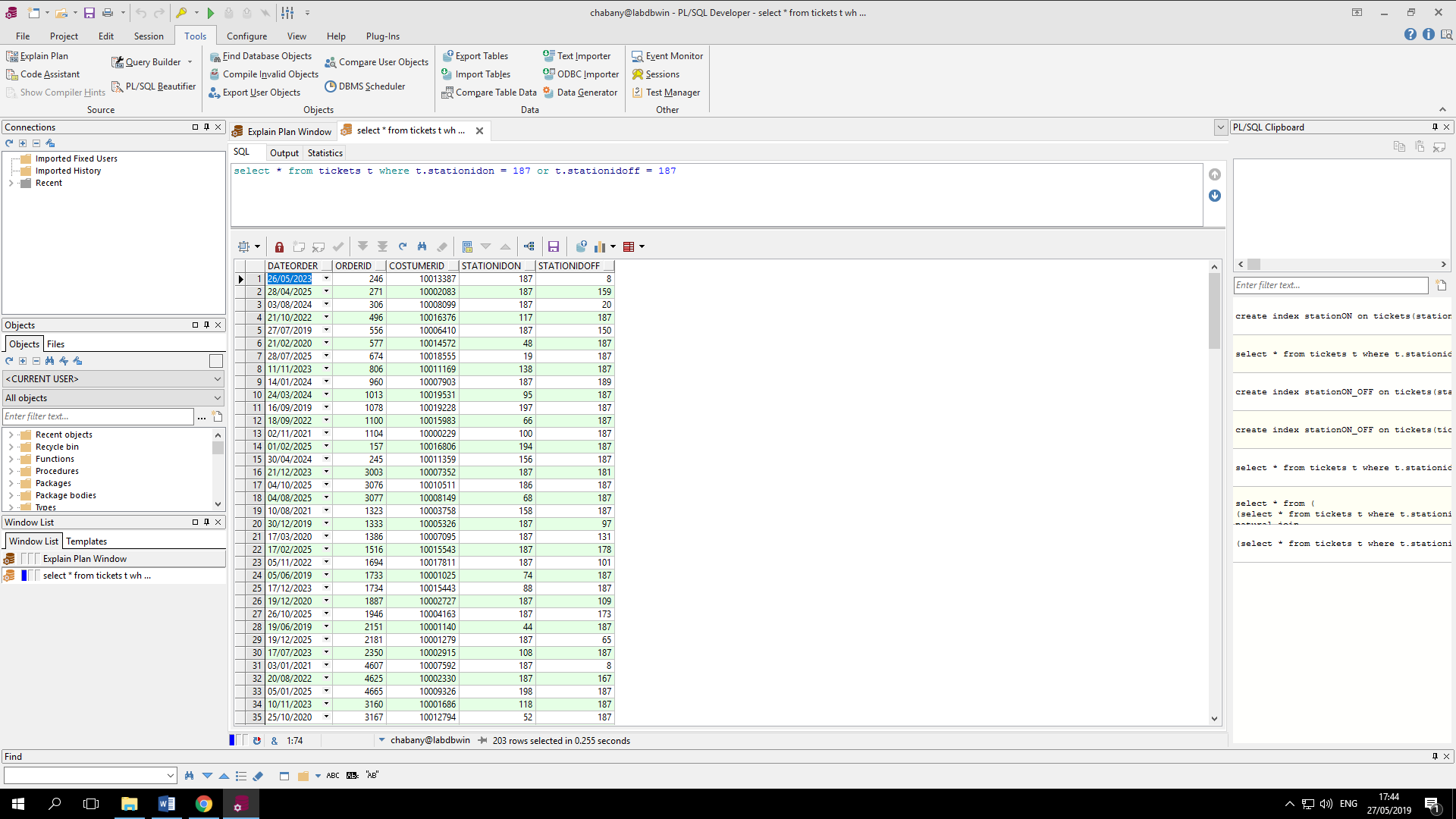


הוספנו עוד 2 אינדקסים עבור כל עמודה בפרד:

create index stationON on tickets(stationidon)

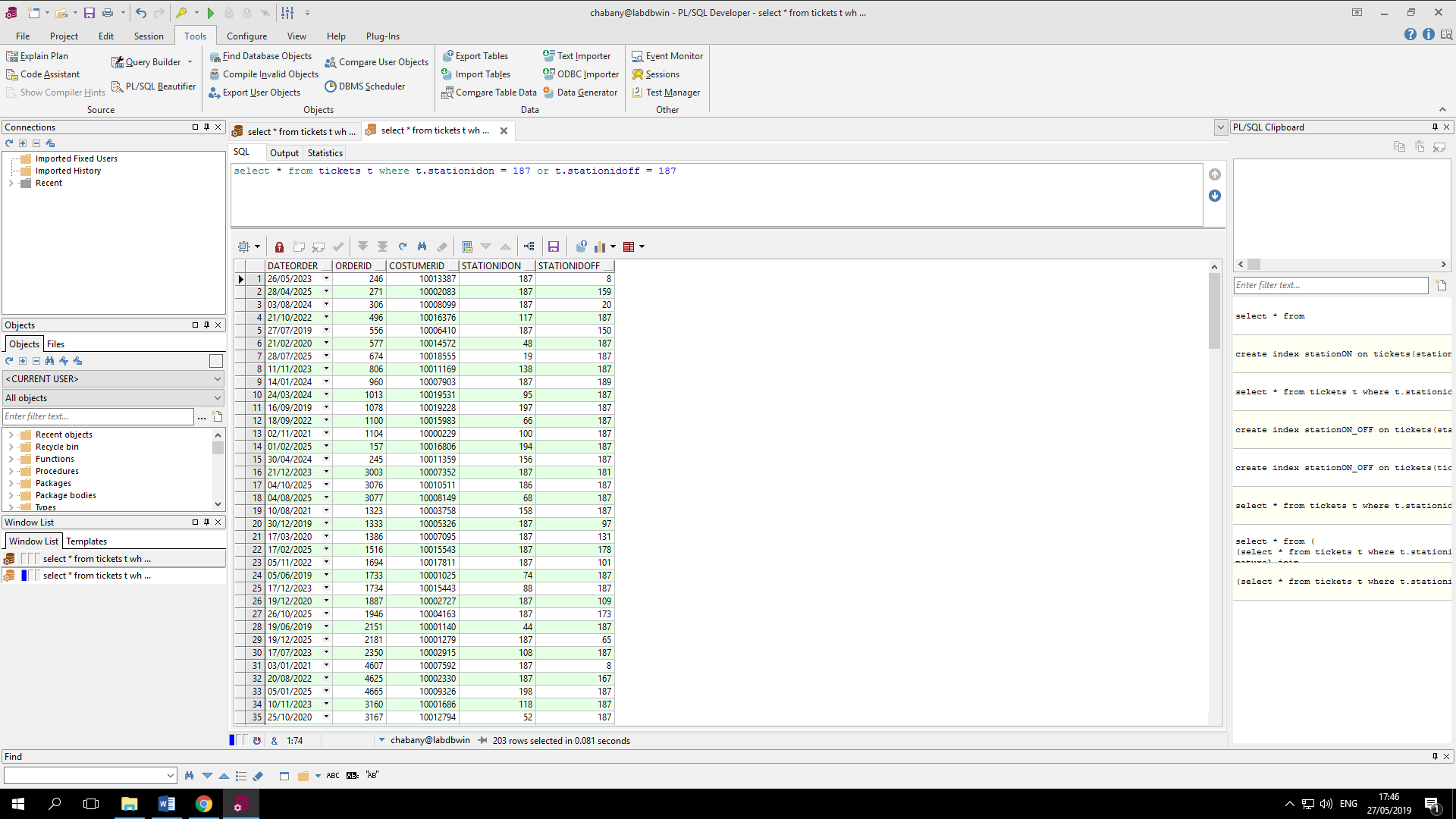
create index stationOFF on tickets(stationidoff)

יצרנו מעמסה על החיפוש ככל הנראה משום שהוא מחפש עכשיו ב3 אינדקבים ולכן התוצאה הייתה גרועה יותר:



עם שלושה אינדקסים

לאחר מחיקת האינדקס של 2 השורות ביחד (עם 2 אינדקסים עבור כל עמודה) התוצאה הייתה מספקת, ככל הנראה משום שבהתחלה היה לו אינדקס רק לעמודה אחת ולכן השינוי לא היה כל כך משמעותי, אך לאחר שחיפשנו שהוספנו גם אינדקס לעמודה השניה, קיבלנו תוצאה מהירה יותר כאמור:



**views**

ניצור view המציג את רשימות הערים בהתאם לשמות התחנות מספרי הערים ומספרי התחנות עם מספרי האיזור

create view station\_view As

select c.areaid as areadid,c.cityid as cityid, c.cityname as cityname ,c\_s.stationid as stationid, c\_s.stationname as stationname from (

city c

inner join

(select \* from city\_stations) c\_s

on c\_s.cityid = c.cityid

)

create view complain\_stationon\_view As (

select \* from (

(select cse.orderid as cseID from COSTUMERSERVICE cse where cse.type = 'complain' )

inner join

(select t.orderId,t.stationIdOn from tickets t) s

on cseid = s.orderid((

נממש שוב את השאילתא של מציאת העיר עם הלקוחות הכי מתלוננים ביחד עם הviews :

select cityname, count(cityid) as complaines from

(select cityname ,cityid from(

(select distinct cseid as compplainid, stationidon from COMPLAIN\_STATIONON\_VIEW) t1

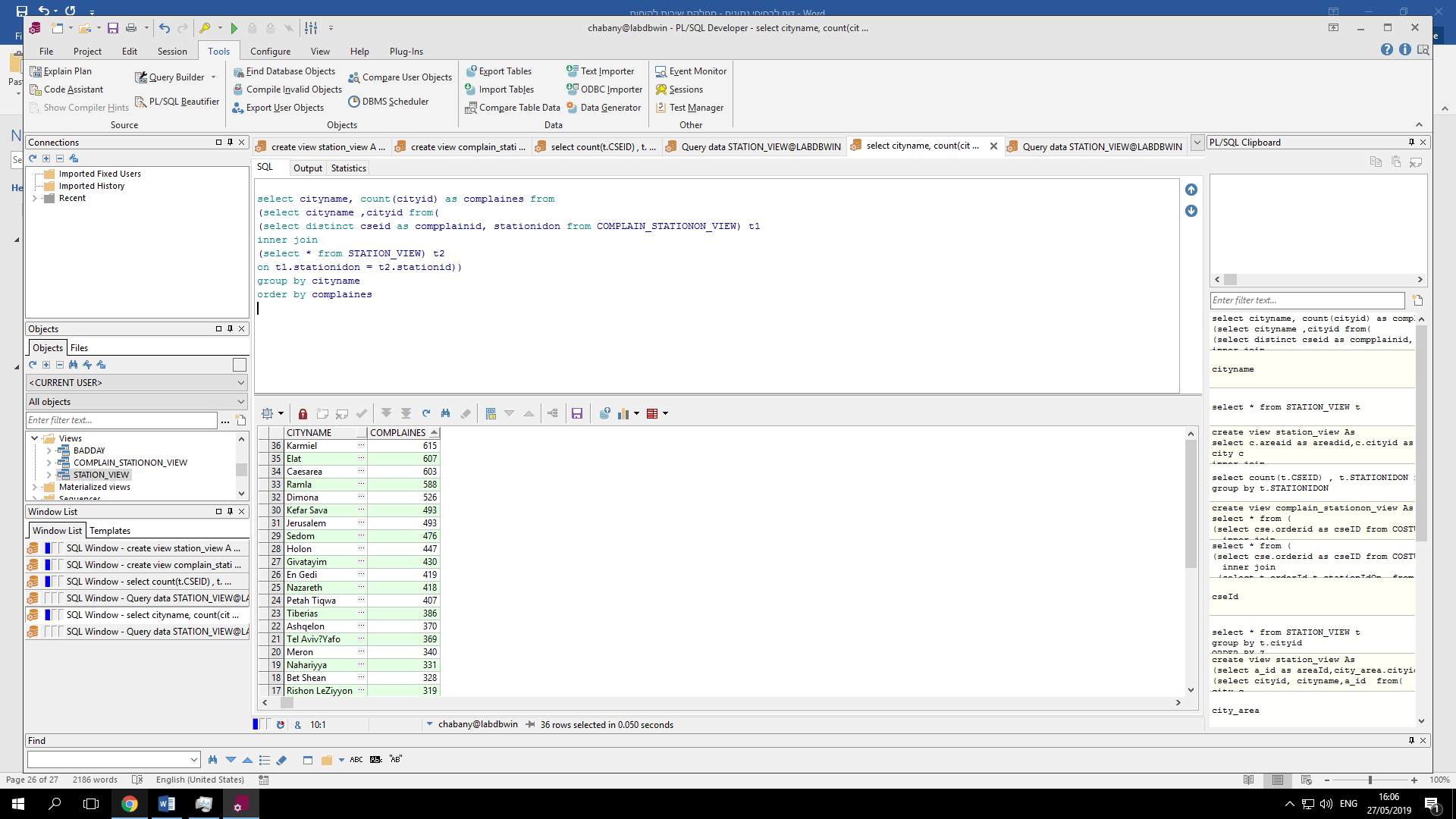
inner join

(select \* from STATION\_VIEW) t2

on t1.stationidon = t2.stationid))

group by cityname

order by complaines



Functions

**פונקציות**

create or replace function find\_costumer(id1 in int) return varchar is

costumer\_name varchar(40);

begin

select distinct FRISTNAME

into costumer\_name

from Costumers

where costumers.costumerid=id1;

return costumer\_name;

end find\_costumer;

הפונקציה שבנינו מקבלת בארגומנט מספר זהות של לקוח ומחזירה את שמו הפרטי.

קטע הקוד :

כדי להריץ את הפונקציה נצטרך לפתוח חלון ולהזין את הקריאה לפונקציה:

DECLARE

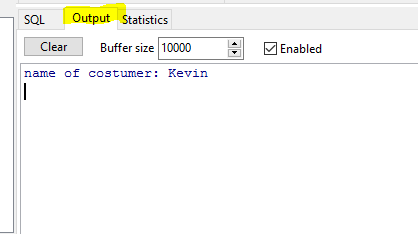
name1 varchar(40);

BEGIN

name1 :=Find\_Costumer(10000223);

dbms\_output.put\_line('name of costumer: ' || name1);

END;

והפלט של הקוד יחכה לנו בלשונית של output:

פונקציה נוספות:

לאחר תלונות רבות של לקוחות שעלו בתחנה 125, עלתה טענה כי העליה בתחנה היא מעל הממוצעים: נכתוב פונצקיה שתחזיר true/false אם העליה בתחנה היא מעל הממוצע או שהיא מתחת לממוצע:

CREATE OR REPLACE FUNCTION gtAvg

RETURN boolean IS

gtavg boolean := true;

BEGIN

SELECT count(\*) into total

FROM customers;

RETURN total;

END;

select stationid , count(stationid) as c from (

(select \* from STATION\_VIEW ) sv

inner join

(select \* from tickets ) t

on sv.stationid = t.stationidon)

group by (stationid)

select stationid , count(stationid) as c from (

(select \* from STATION\_VIEW ) sv

inner join

(select \* from tickets ) t

on sv.stationid = t.stationidon)

group by (stationid)

## הרשאות

כדי שנוכל לעבוד ככיתה שלמה המפתחת בסיס נתונים כאשר כל קבוצה בונה טבלאות שונות לבסיס הנתונים ואנו צריכים לגשת לטבלאות שלהם כדי לקחת מידע אנו צריכים לקבל מיוצרי הטבלאות הרשאה כדי שנוכל להשתמש בטבלאות שלהם.

במקרה שלנו כמעט ולא היינו צריכים לתת הרשאות לאף אחד מהסיבה הפשוטה שאנחנו היינו די תלויים באחרים מכיוון שבמתקנים השייכים למלונות מסוימים יש התייחסות גם לעובדים גם ללקוחות וגם למוצרים.

Grant select on costumerservice to ytzadik