**בס"ד**

**מיני פרויקט בבסיסי נתונים**

גרסה: 1.6

תאריך: 4 באוגוסט 2020

מגישים: ראובן לריאונוב 211996269, יוסף קטרי 322263310

קישור לgithub: <https://github.com/yossefkatri/150225-5779-Databases>

קורס: מיני פרויקט בבסיסי נתונים(150225)

מספר קבוצה: 42

תוכן עניינים

[מבוא 2](#_Toc47443776)

[עבודת הכנה והכרת התוכנה 2](#_Toc47443777)

[תרשים ERD 2](#_Toc47443778)

[תיאור הישויות והקשרים 2](#_Toc47443779)

[ישויות 2](#_Toc47443780)

[קשרים 3](#_Toc47443781)

[נרמול הטבלאות 3](#_Toc47443782)

[תרשים DSD 3](#_Toc47443783)

[הפרויקט שלנו 4](#_Toc47443784)

[תרשים ERD 4](#_Toc47443785)

[תיאור הישויות והקשרים 4](#_Toc47443786)

[ישויות 4](#_Toc47443787)

[קשרים 5](#_Toc47443788)

[תרשים DSD 6](#_Toc47443789)

[יצירת הטבלאות 6](#_Toc47443790)

[הכנסת נתונים 6](#_Toc47443791)

[שאילתות SQL 8](#_Toc47443792)

[בחירה - SELECT 8](#_Toc47443793)

[אינדקסים 12](#_Toc47443794)

[אינטגרציה 16](#_Toc47443795)

[שאילתות האינטגרציה 17](#_Toc47443796)

[תרשימים 18](#_Toc47443797)

[Views 19](#_Toc47443798)

[פונקציות 23](#_Toc47443799)

[פרוצדורות 24](#_Toc47443800)

[נספחים 26](#_Toc47443801)

[נספח ראשון: יצירת הטבלאות 26](#_Toc47443802)

[נספח שני: שאילתות ואינדקסים 27](#_Toc47443803)

[נספח שלישי: Integration 28](#_Toc47443804)

[נספח רביעי: views 28](#_Toc47443805)

[נספח חמישי: פונקציות ופרוצדורות 29](#_Toc47443806)

# מבוא

* **מחלקת מתנדבים ואמבולסים**

בתי חולים: מספר מזהה, שם, כתובת, קיבולת.

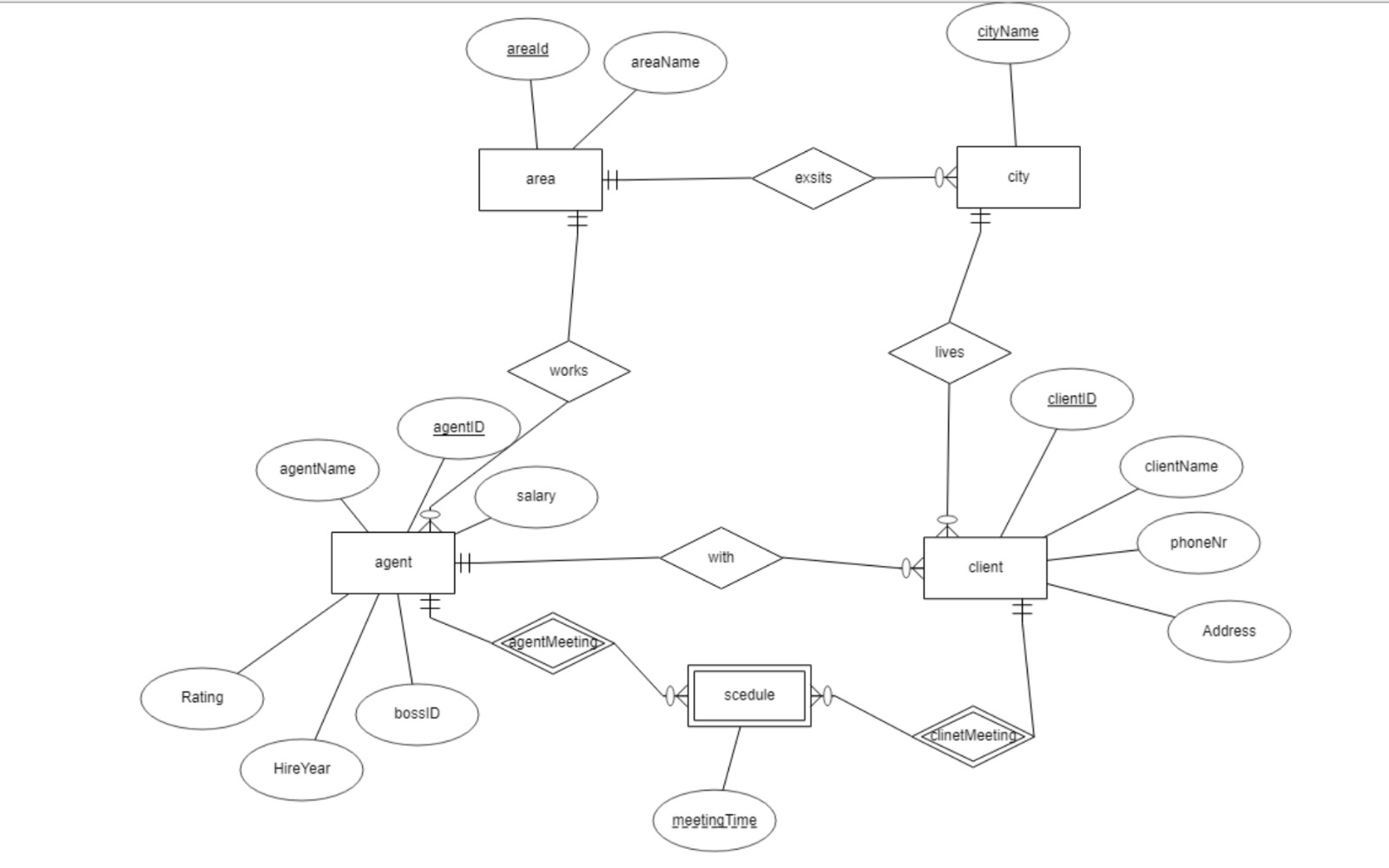
מתנדבים: מספר מזהה, שם, כתובת, מספר מזהה בית חולים, מספר פלאפון, תפקיד.

חברות אמבולנס: מספר מזהה, שם, מס' טלפון, כתובת.

אמבולנסים: מספר לוחית רישוי, פלאפון, מספר מזהה בית חולים, מספר מזהה חברה.

# 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.

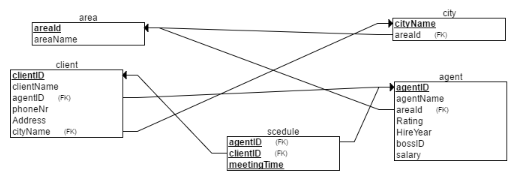
### קשרים

* 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, bossID, salary, areaID)
* Client (clientID, clientName, address, phoneNr, cityName)
* CityName (cityName, areaID)
* Area (areaID, areaName)

## תרשים DSD

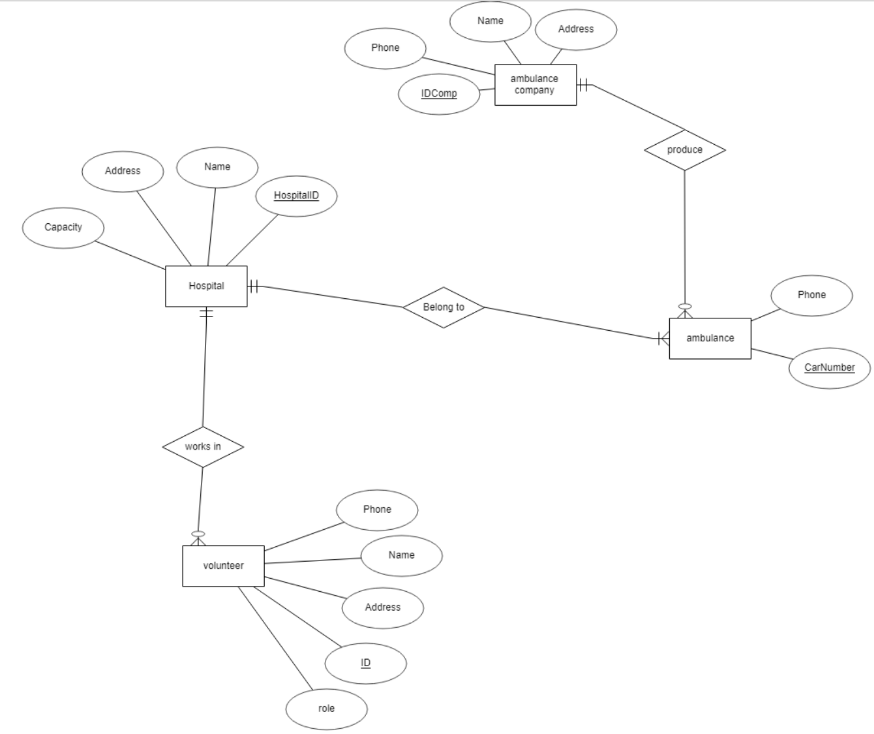


# הפרויקט שלנו

בקבוצתנו החלטנו לתרום את כוחנו למען המאבק במגפת הקורונה בפרט ובמצבי חירום בכלל. אנו החלטנו ליצור מערכת ובסיס נתונים של כל המערך הרפואי המעורב בפיתרון מצבי חירום כגון מגפות, רעידות אדמה ומלחמות. לקחנו על עצמנו את אחריות הטיפול בתחום המתנדבים והאמבולנסים בבתי החולים. תפקיד המערכת הזאת כחלק מהמערך הרפואי הוא לארגן את המתנדבים והאמבולנסים שנמצאים בבתי החולים, ולספק במידת הצורך אמבולנסים ועובדים לבתי החולים השונים.

## תרשים ERD

כפי שהזכרנו, במחלקה שלנו, ישנן 4 ישויות: בתי חולים, מתנדבים, חברות אמבולנסים ואמבולנסים. בשלב הראשון יצרנו תרשים ERD שיתאר את הקשרים בין הישויות הללו ואת התכונות שלהן.



## תיאור הישויות והקשרים

### ישויות

* Hospitals – ישות זאת אחראית על כל הבתי חולים הנמצאים במערכת.
* Hospitals\_Id – מספר מזהה של הבית חולים (PK).
* Hospitals\_Name – שם הבית חולים.
* Capacity – תפוסה בבית חולים.
* Address – כתובת הבית חולים.
* Volunteers – ישות זאת אחראית על כל המתנדבים הנמצאים במערכת.
* Volunteers\_Id – מספר מזהה של המתנדב (PK).
* Volunteers\_Name – שם המתנדב.
* Hosptal\_id – מספר מזהה של בית חולים (FK).
* Address – תאריך בקשת האספקה.
* Phone – מס' פלאפון.
* Ambulance\_Company – ישות זאת אחראית על כל החשבונות הנמצאים במערכת.
* Ambulance\_Company\_id – מספר מזהה של חברת האמבולנסים (PK).
* Ambulance\_Company\_Name – שם חברת האמבולנסים.
* Address – כתובת.
* Phone – מס' טלפון.
* Ambulance - ישות זאת אחראית על כל האמבולנסים הנמצאים במערכת.
* Ambulance\_Id – מספר מזהה של המתנדב (PK).
* Hosptal\_id – מספר מזהה של בית חולים (FK).
* Ambulance\_Company\_id – מספר מזהה של חברת אמבולנסים (FK).
* Phone – מס' פלאפון.

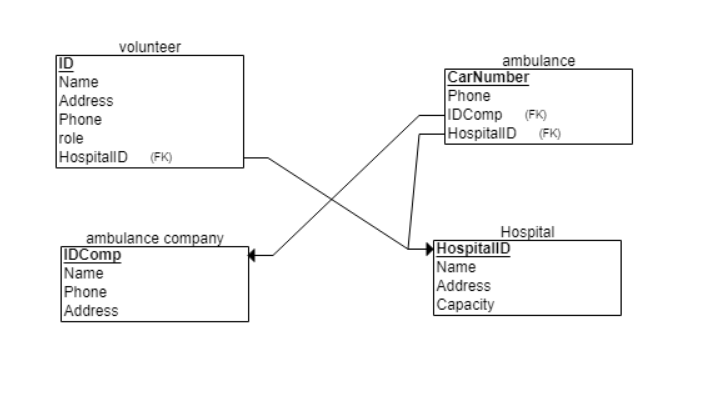
### קשרים

* Produce – הקשר בין Ambulance לבין Ambulance\_Company. הקשר הוא 1:M משום שחברת אמבולנסים אחת יכולה לייצר הרבה אמבונסים, אבל אבל לכל אמבולנס יש רק חברה אחת שייצרה אותו.
* Belong to – הקשר בין Ambulance לבין Hospitals. הקשר הוא M:1 משום שיכול להיות לבית חולים אחד הרבה אמבולנסים, אבל אמבולנס אחד שייכת לבית חולים אחד בלבד.
* Works in – הקשר בין Volunteers לבין Hospitals. הקשר הוא M:1 משום שיכול להיות לבית חולים אחד הרבה מתנדבים, אבל מתנדב אחד מתנדב בבית חולים אחד בלבד.

## תרשים DSD

* על פי תרשים ה- ERD ועל ידי הבנת הקשרים בין הישויות, יצרנו תרשים DSD עבור החלק שלנו במערכת: מחלקת מתנדבים ואמבולסים

.



## 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.

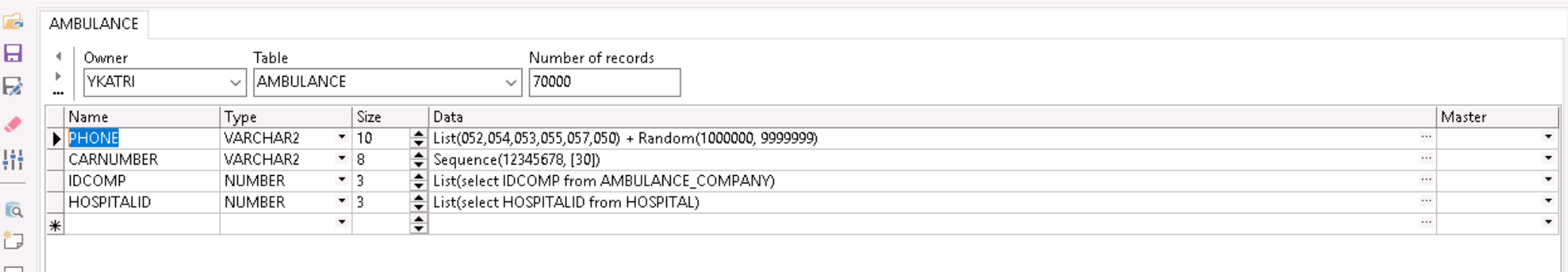
יצרנו קוד לייצור הטבלאות באמצעות export SQL של האתר erdPlus, יצרנו קובץ SQL ואז העתקנו את קוד ה- SQL של כל טבלה אל תוכנת ה-plsql לשם יצירת הטבלאות בפועל(הקוד של הטבלאות נמצא בנספח).

## הכנסת נתונים

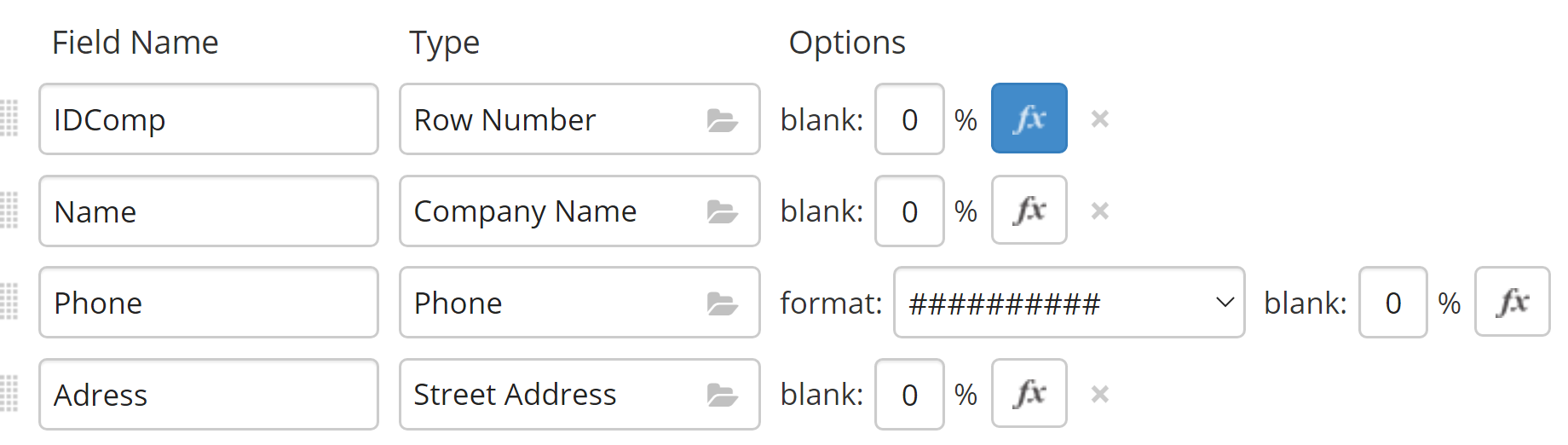
השתמשנו במגוון כלים כדי לאכלס את הטבלאות. הכלי העיקרי שהשתמשנו הוא data generator בתוכנת ה- PL/SQL כדי להכניס את הנתונים לשניים מתוך ארבעת הסכמות (אמבולנסים ומתנדבים).הכלי הזה מאפשר לך יצירה מהירה של נתונים רבים ולייצר נתונים בעזרת סכמות קיימות (אפשר לראות דוגמא לכך בתמונה הראשונה).

כלי נוסף שהשתמשנו הוא באתר mockaroo על מנת למלא את טבלת חברות האמבולנסים ואת טבלת בתי החולים. האתר מאפשר בחירה מגוונת של נתוני סרק. באתר mockaroo אפשר להוריד את הנתונים כקובץ של פקודות INSERT או כקובץ csv. עד אלף נתונים (כמנוי חינמי).( אפשר גם לייצר נתונים בעזרת סכמות קיימות אבל אנחנו לא השתמשנו בזה.)

כך לדוגמא, הכנסת נתונים בעזרת data generatorלסכמה של האמבולנסים:



וכך לדוגמא, הגדרנו איך לחולל את הנתונים של הטבלה של החברות אמבולנסים:



והורדנו את הנתונים של הטבלה לקובץ sql כזה:



ואילו לסכמה של בתי החולים הורדנו את הנתונים כקובץ csv כזה:



כמות הנתונים:

* אמבולנסים- 70,000
* מתנדבים- 20,000
* בתי חולים- 500
* חברות אמבולנסים- 400

נתקלנו בבעיה אחת, כשרצינו להכניס נתונים לכל אחד מהטבלאות יש ערך UNIQE ,כלומר היינו צריכים לדאוג שהמספרים בערך הזה, לא יחזרו על עצמם, כדי לפתור את הבעיה השתמשנו במספר סידרתי שעולה עם כל נתון שאנחנו מכניסים, כך שלכל נתון יש ערך ייחודי באותו שדה.

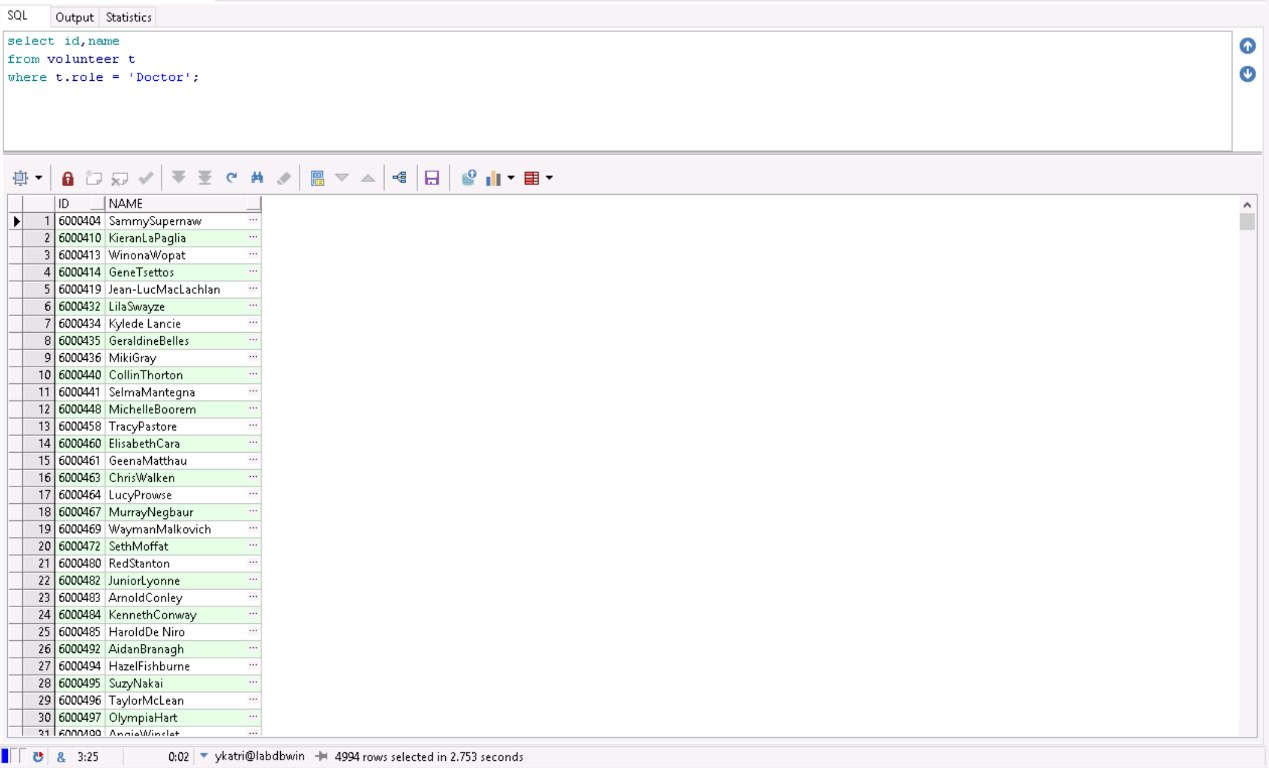
בנוסף, היו זוגות נוספים שמימשו את הישות בית חולים וגם הם התייחסו למספרי זהות של אנשים וכדי למנוע כפילות חילקנו את מספרים שישמשו כת"ז בין כולנו, אנחנו השתמשנו ב6000000-7000000.

## שאילתות SQL

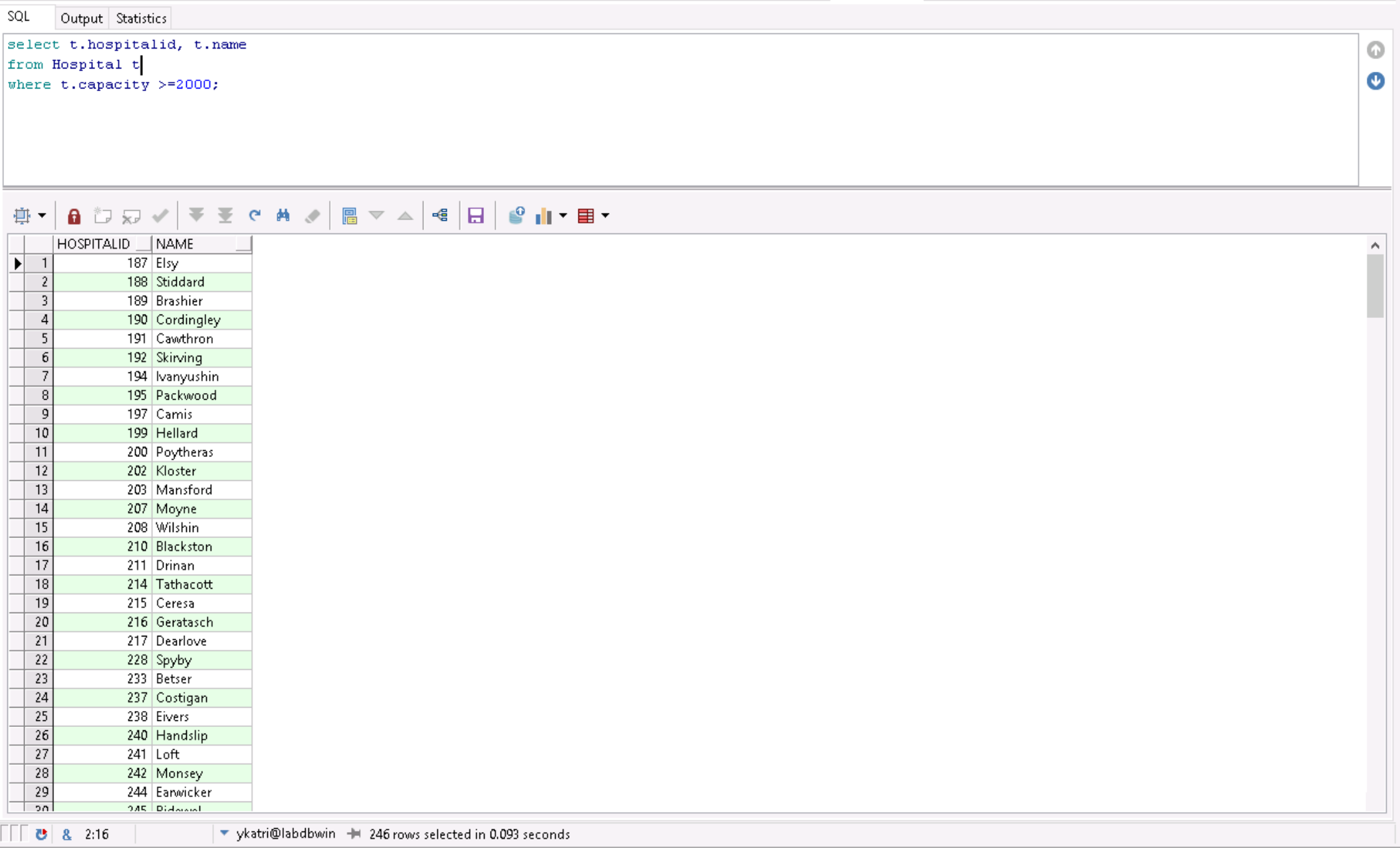
### בחירה - SELECT

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

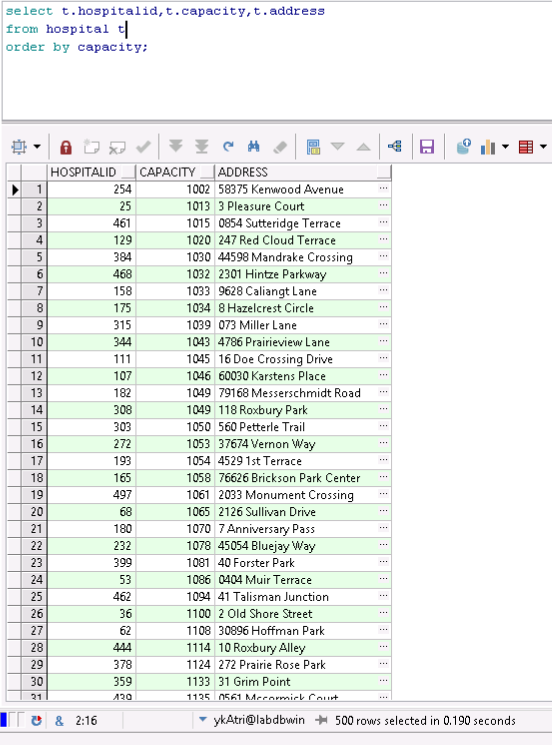
1. כחלק מהרצון לניהול נתונים ומשאבים נוח אנו כתבנו שאילתא שמחזירה את השם ותעודת הזהות של כל הרופאים המתנדבים.



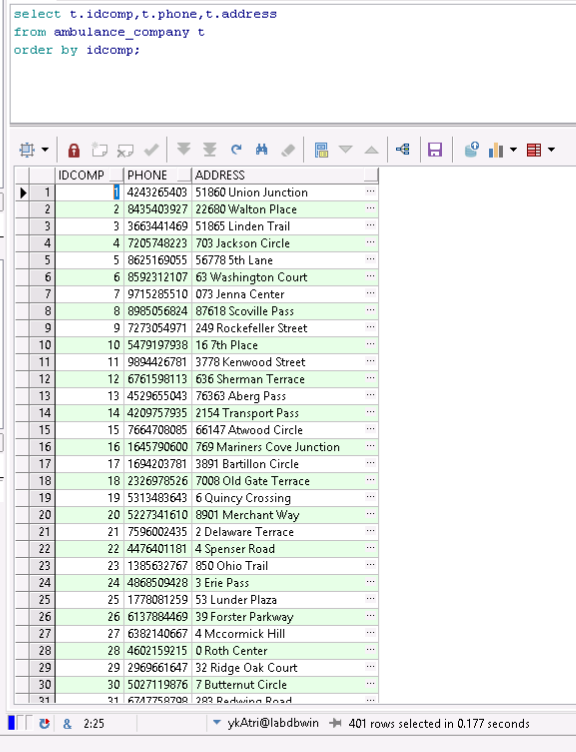
1. במידה ויהיה אסון גדול כלשהו או מרכז הדבקה גדול ירצו לשלוח את הנפגעים לבית חולים גדול, ולכן יצרנו שאילתא שמחזירה את השם והמספר הסידורי של כל בתי החולים שיש להם קיבולת גדולה מ2000.



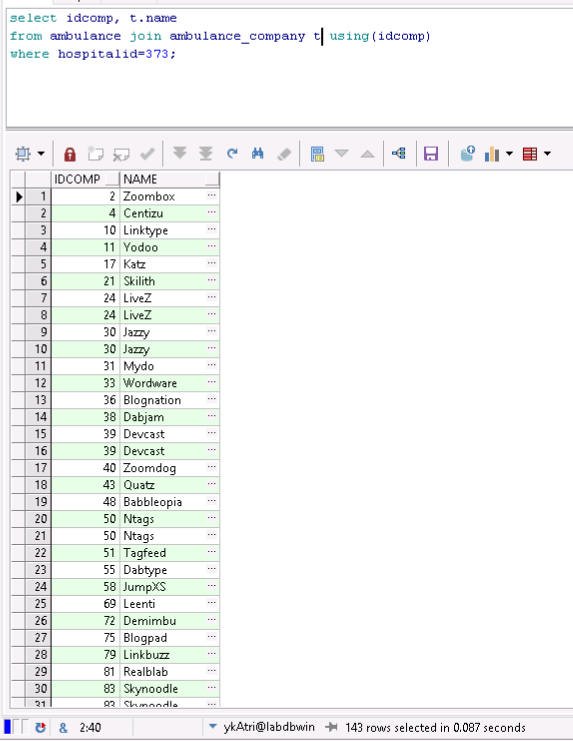
1. במידה והממשלה תרצה להרחיב בית חולים מסוים אנו נרצה גם לדעת מה בית החולים בעל הקיבולת הקטנה ביותר ואת המיקום שלו, אז יצרנו שאילתא שתחזיר לנו את רשימת המספר הסידורי, הקיבולת של בתי החולים והכתובת ממוינים על פי הקיבולת.



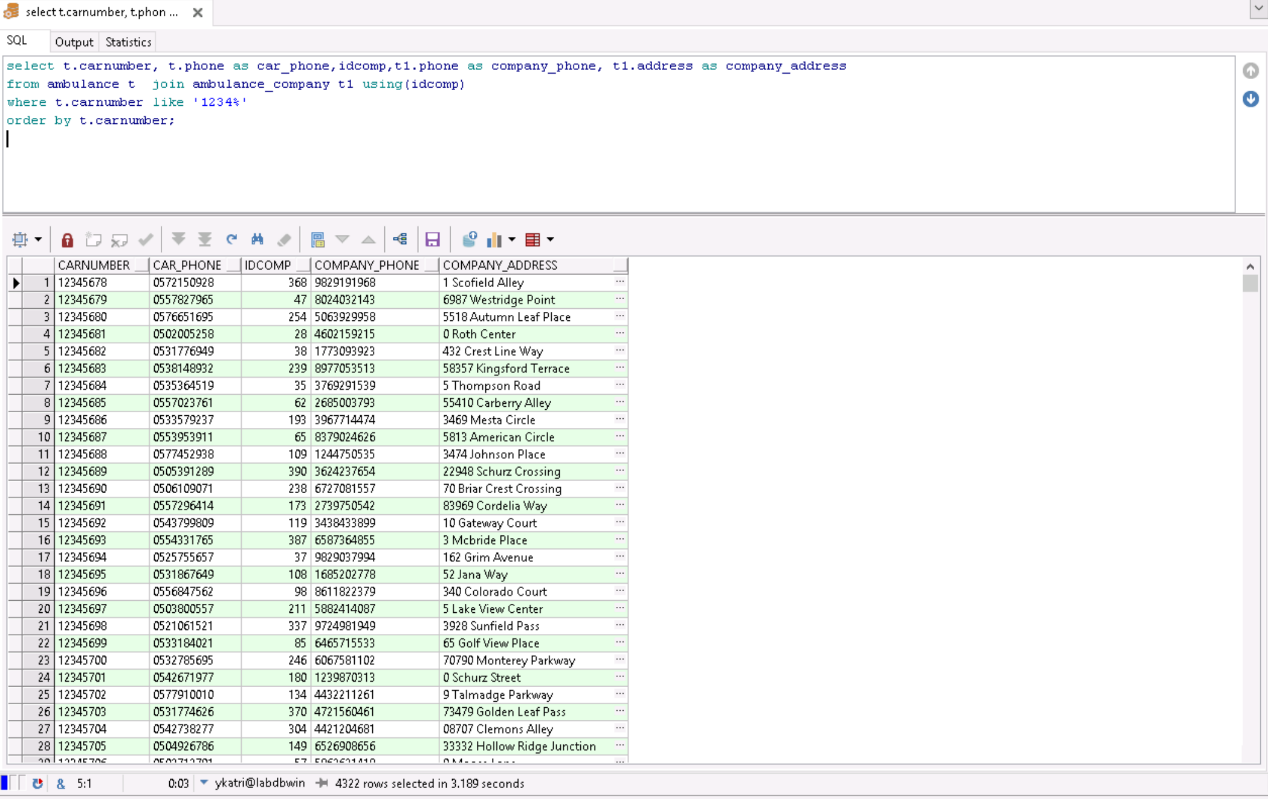
1. במקרה שאמבולנס זקוק לתיקון נרצה לאתר בקלות את חברת האמבולנסים שיצרה אותו על מנת שנוכל לשלוח אותו אליה, לכן יצרנו שאילתא שמחזירה את חברות האמבולנסים ממוינות ע"פ מ"ס הזיהוי שלהם, יחד עם הטלפון והכתובת שלהם.



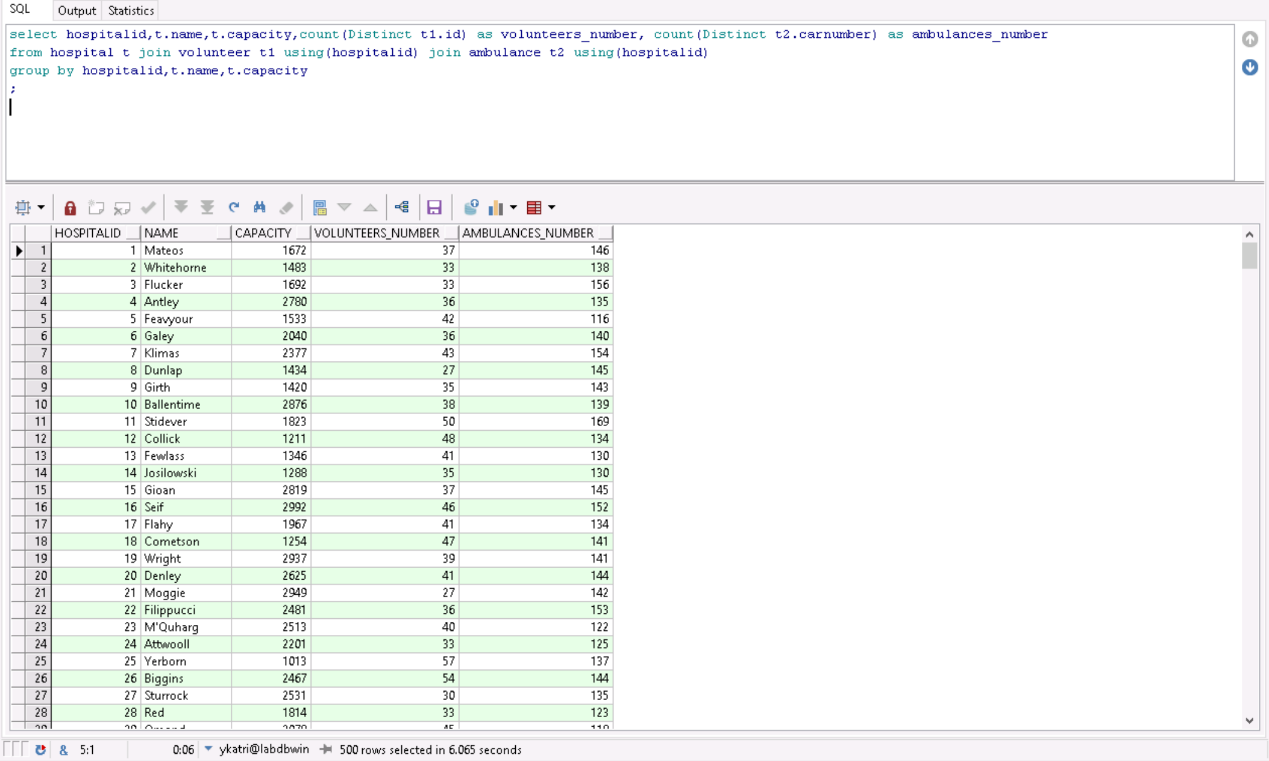
1. בית החולים מעוניין לדעת את כל חברות האמבולנסים שעובדות איתו מסיבות אדמיניסטרטיביות, לכן יצרנו שאילתא שמחזירה את חברות האמבולנסים של בית חולים ספיציפי.



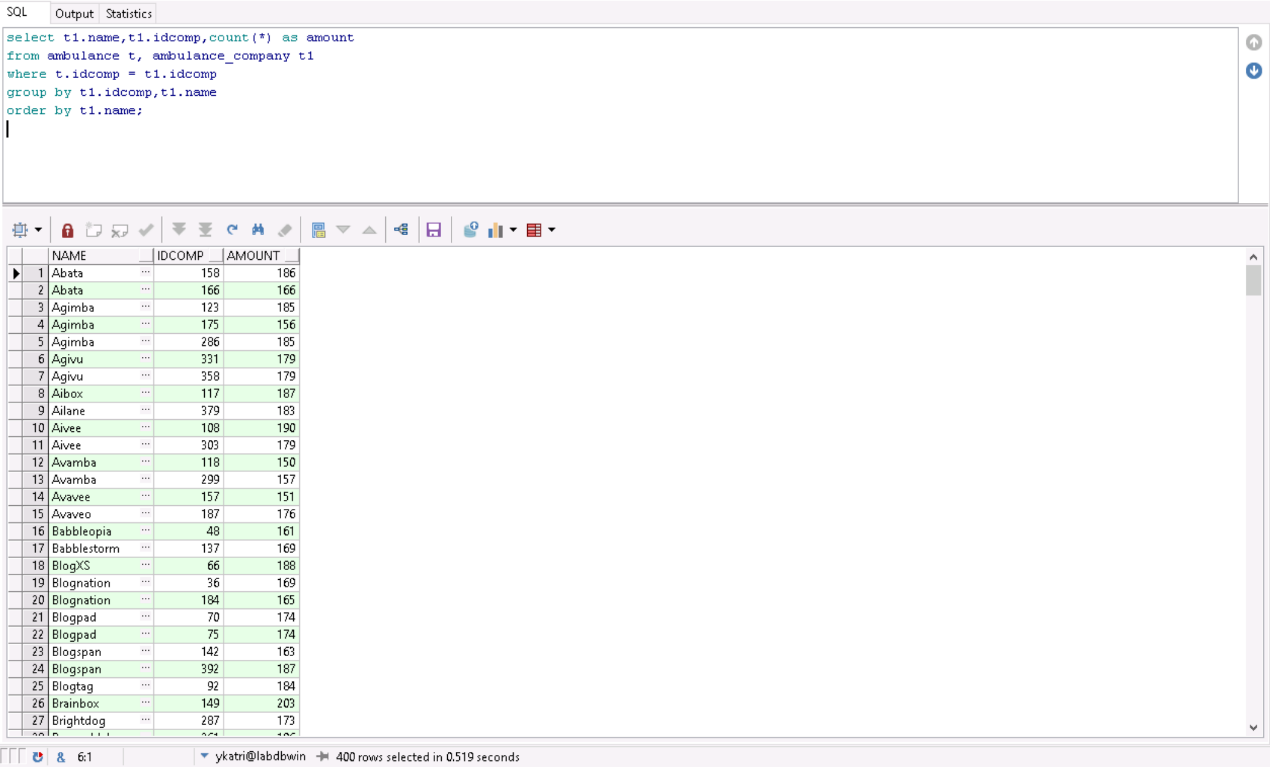
1. במקרה של גניבת אמבולנס על מנת שנוכל לאתר בקלות אמבולנס כלשהו ואת החברה שיצרה אותו כאשר אנחנו יודעים חלק או את כל הלוחית רישוי שלו, יצרנו שאילתא שמחזירה את לוחית הרישוי, מספר הטלפון של האמבולנס, המספר הסידורי של החברה, הטלפון והכתובת שלה במקרה שחלק או כל הלוחית רישוי מתאימה ללוחית הרישוי שחיפשנו.



1. על מנת לקבל דוח שותף על איזון בתי החולים בין כמות האמבולנסים לכמות המתנדבים והקיבולת, יצרנו שאילתא שמרכזת את כל הנתונים האלה עבור כל בית חולים.



1. על מנת להקל על רשות המס את עבודתה יצרנו שאילתא שתחזיר לנו את המספר הסידורי והשם של כל חברות האמבולנס ואת מספר האמבולנסים שהן יצרו.



## 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.

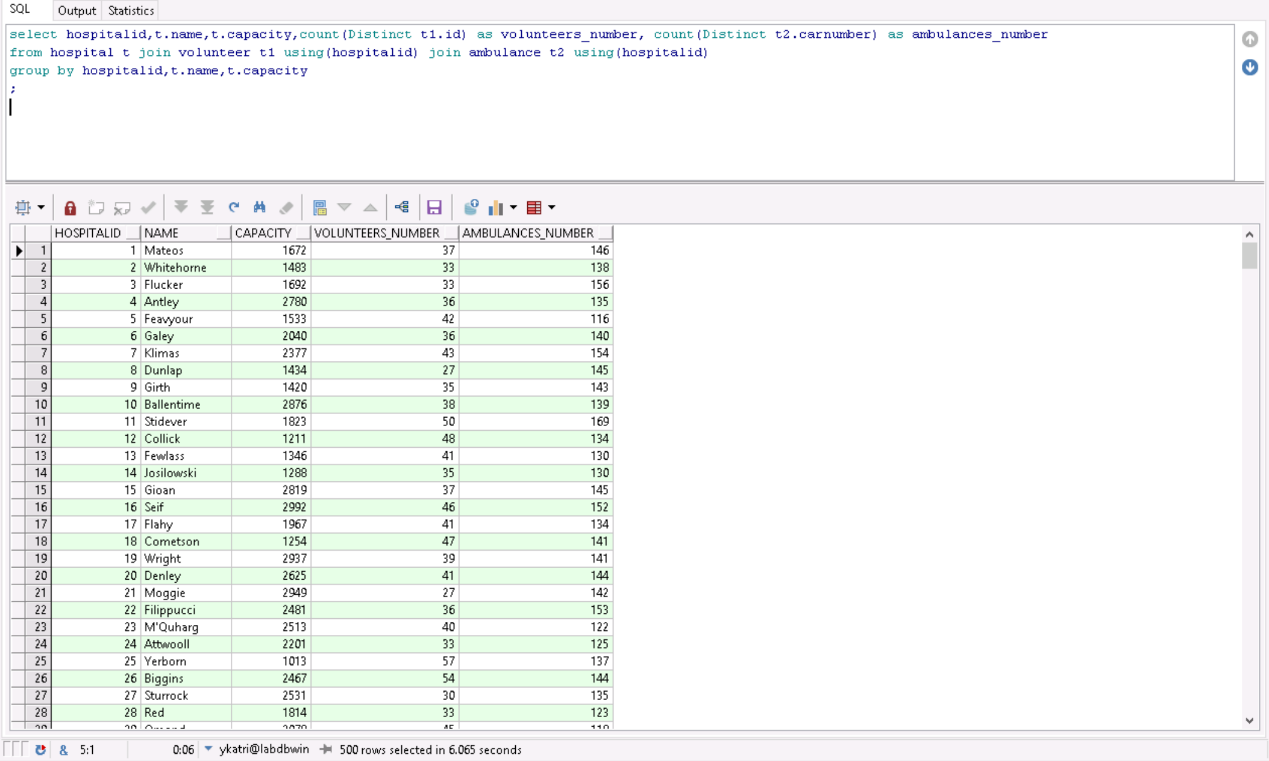
במידה ולא נגדיר אינדקס לטבלה אז בכל שאילתה על הטבלה השאילתה תגרום למעבר על כל הרשומות בטבלה עד שתמצא את כל הרשומות העונות למה שחיפשנו. כשנגדיר אינדקס מתאים אז החיפוש יהיה מהיר יותר כי הפניה לבסיס הנתונים תגרום לזה שמנוע החיפוש בבסיס הנתונים יפנה קודם לאינדקס וילך לרשומות המתאימות על פי מה שרשום באינדקס.

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

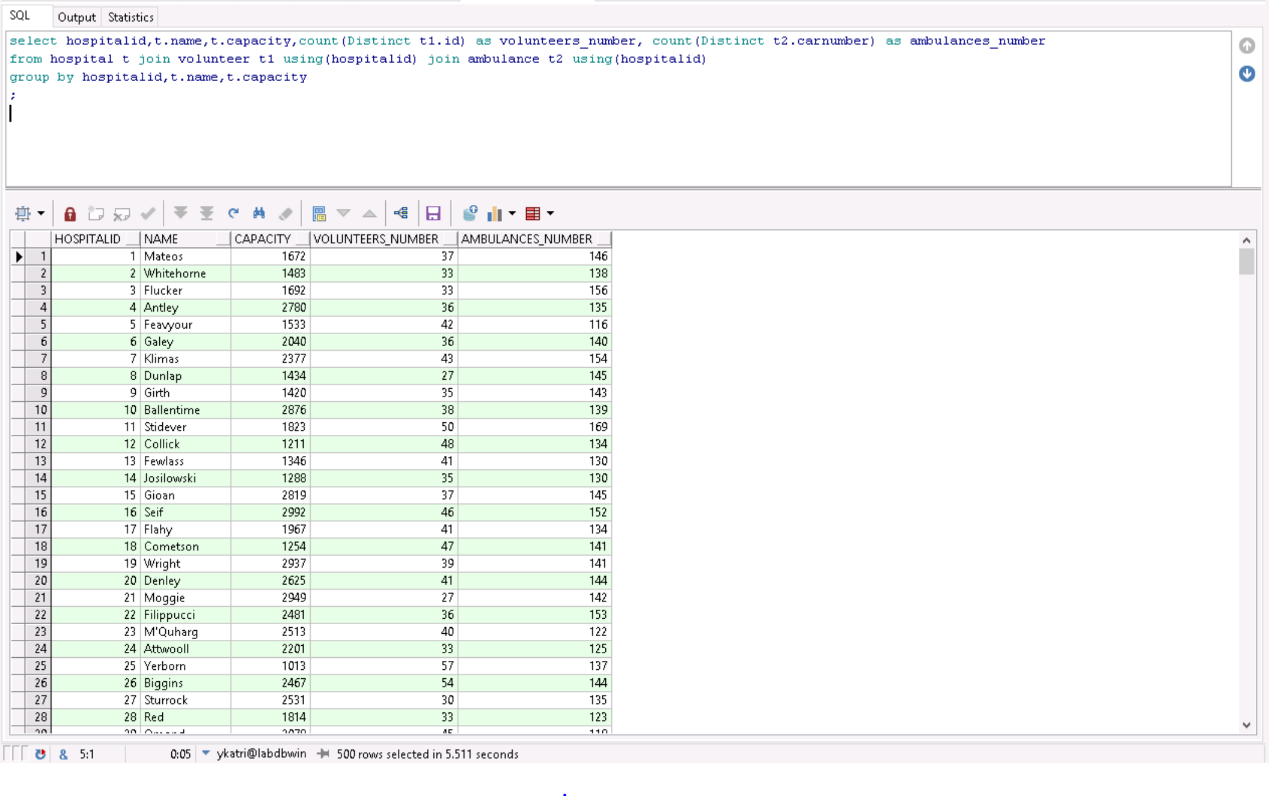
עבור שאילתא מספר 7 יצרנו את האינדקס הבא:

האינדקס ממיין את המפתחות (המספר הסידורי) של הבתי חולים עבור טבלת האמבולנסים. ההסתברות לקבל בית חולים ספיציפי היא כ-0.002.בזכות המיון, האינדקס מקל בשאילתא זאת את ביצוע הJoin בין שתי הטבלאות כתוצאה מכך הוא מקצר את זמן הריצה של השאילתא בכחצי שנייה, זמן זה אינו משמעותי ולכן הוספת האינדקס אינה משמעותית.

לפני:



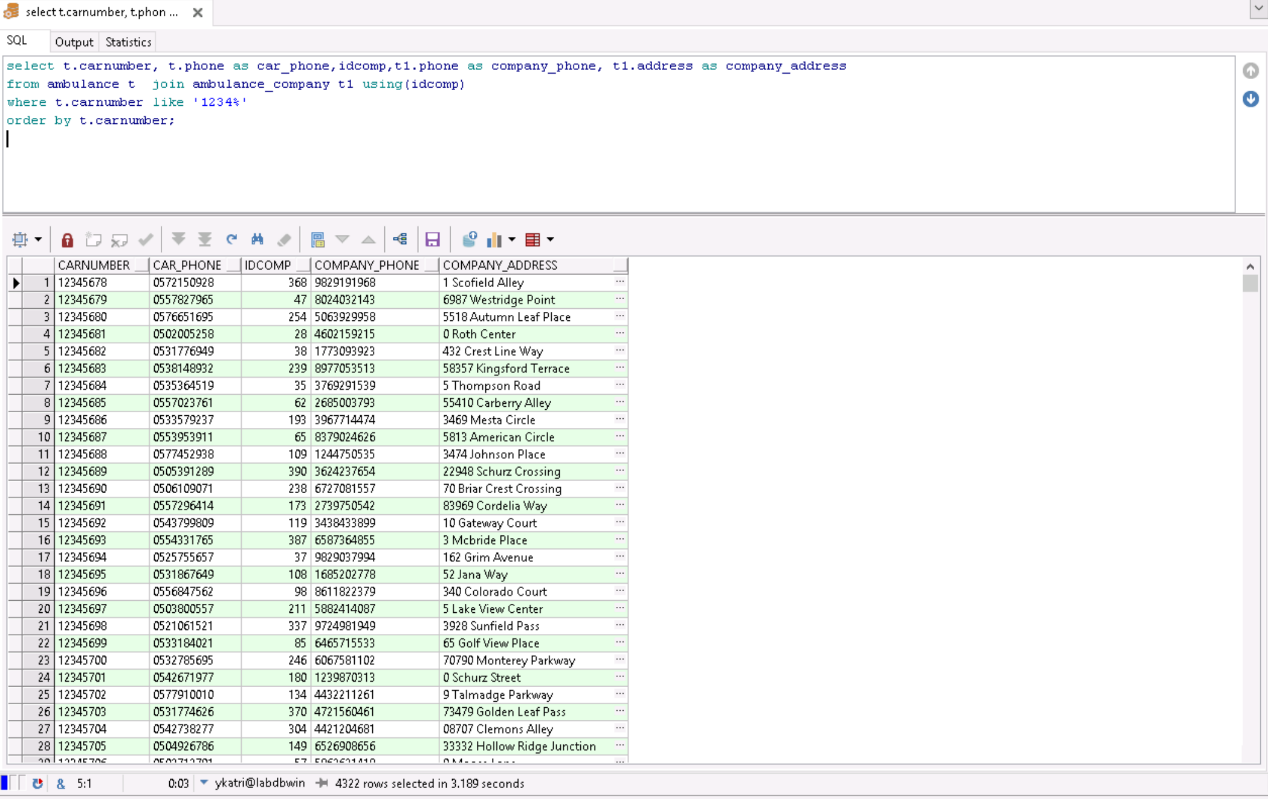
אחרי:



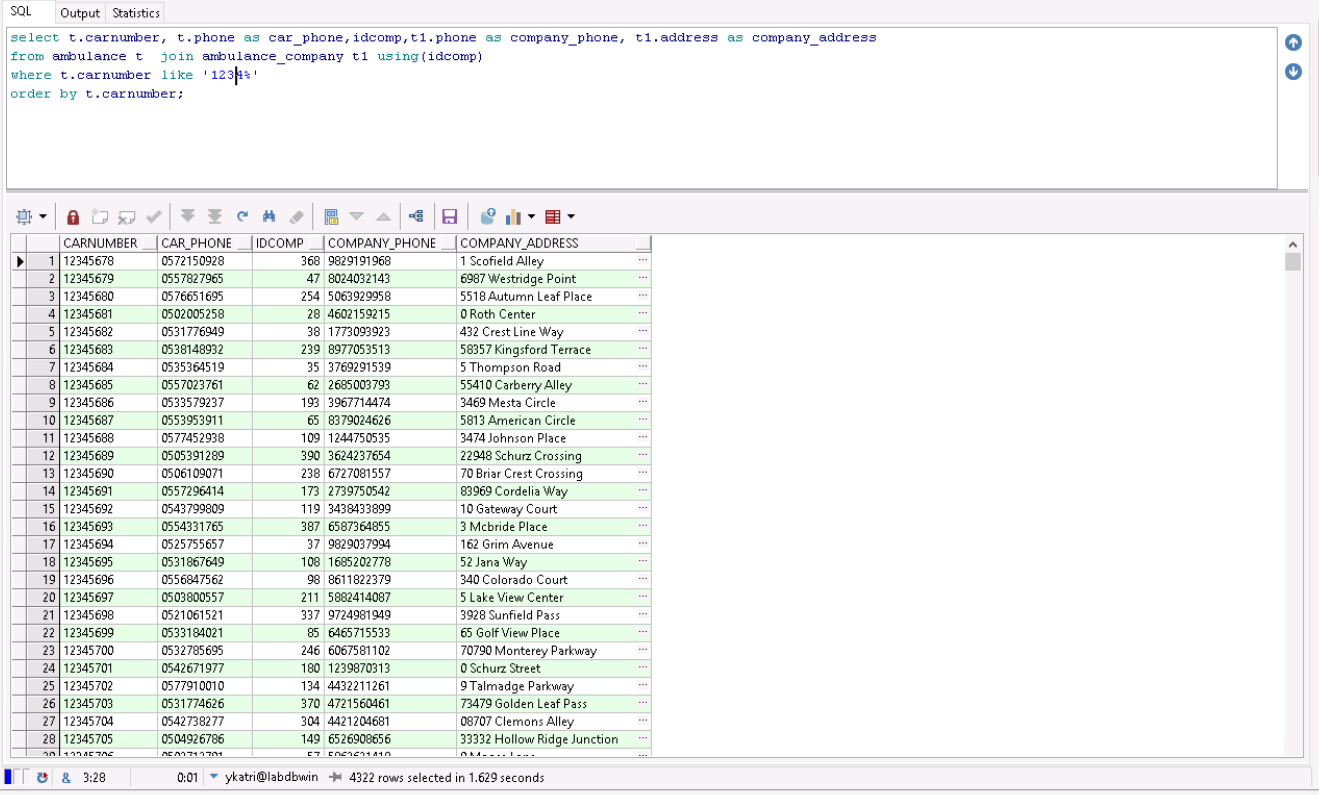
עבור שאילתא מספר 6 יצרנו את האינדקס הבא:

האינדקס ממיין את המפתחות (המספר הסידורי) של החברות האמבולנס עבור טבלת האמבולנסים. ההסתברות לקבל חברה ספציפית היא כ-0.0025. בזכות המיון, האינדקס מקל בשאילתא זאת את ביצוע הJoin בין שתי הטבלאות וכתוצאה מכך הוא מקצר את זמן הריצה של השאילתא, זמן הריצה התקצר בכ-1.5 שניות.

לפני:



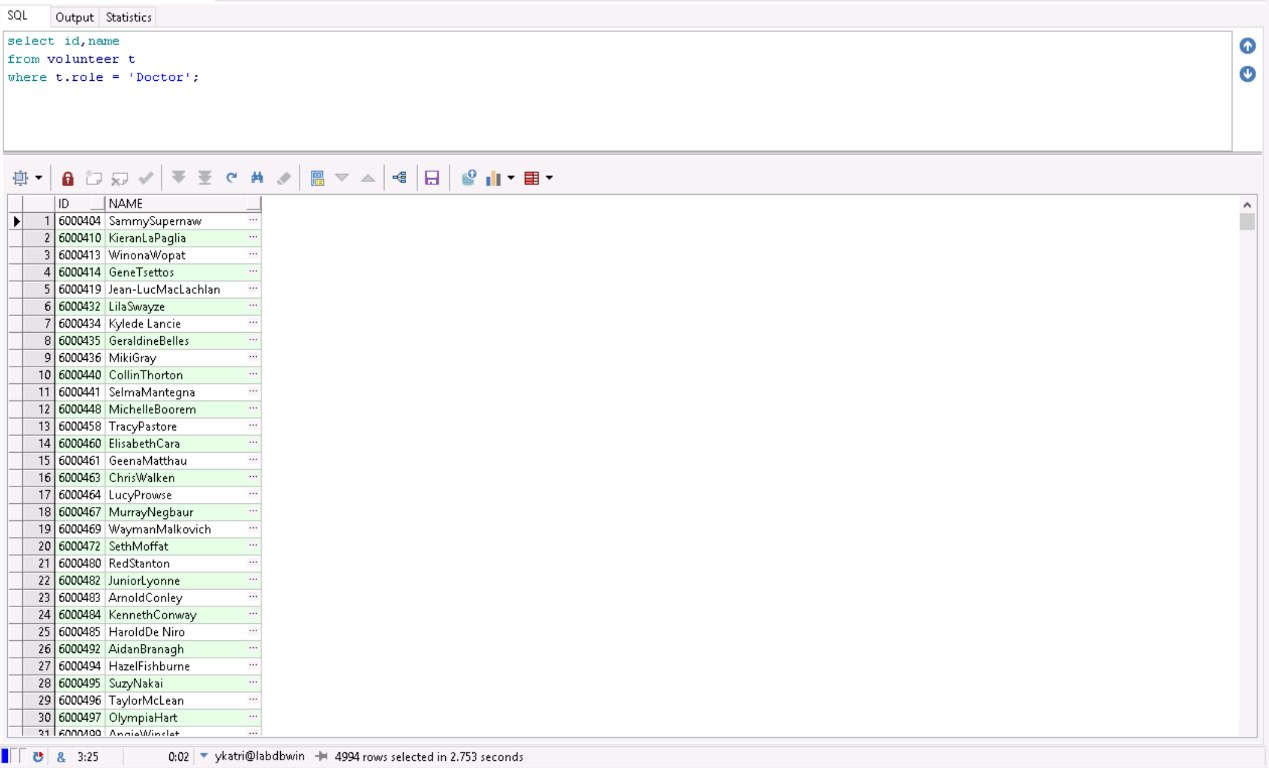
אחרי:



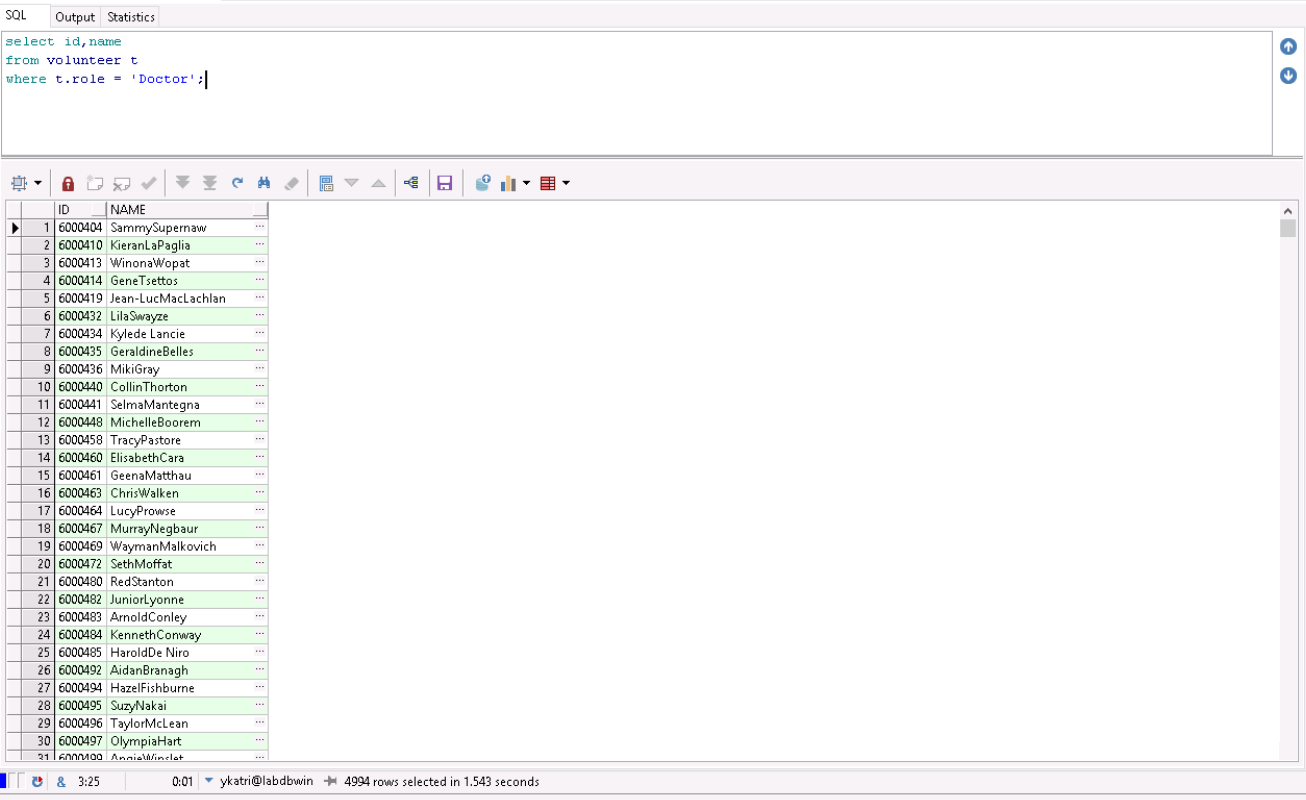
עבור שאילתא 1 יצרנו את האינדקס הבא:

אינדקס שממיין את התפקיד (role) של כל מתנדב (volunteer) עבור טבלת המתנדבים. ההסתברות לקבל שם של תפקיד ספיציפי היא כ-0.25. בזכות המיון, האינדקס מקל בשאילתא על ביצוע ה-where ומקצר את זמן הריצה של השאילתא, זמן הריצה התקצר בכשנייה.

לפני:



אחרי:



## אינטגרציה

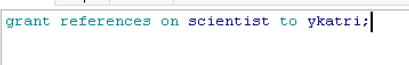
החלטנו שעל מנת לעזור למדענים לפתח פתרון מדעי לבעיה שתצוץ בין אם זה יהיה אסון טבע או מגפה, נצוות להם מתנדב מבית החולים שהם עובדים בוא, על מנת לעשות זאת נקבל הרשאת גישה לישות הScientist מאבינועם ואליעזר.

אולם, בתכנון המקורי של הישויות בבעלותנו וכן בבעלות אבינועם ואליעזר לא היו ישויות הקשורות אחת לשנייה. לכן, אנו ניצור טבלה שמקשרת בין מתנדב למדען שקוראים לה Works. הID של המתנדב יהווה מפתח לטבלה מכיוון שאנחנו רוצים שיהיה לכל מתנדב מדען אחד שהוא יעבוד איתו.

סכמת Works:

* IDVolunteer - ת"ז של מתנדב
* IDScientist - ת"ז של מדען

על מנת שנוכל ליצור סכמה שעושה REFERENCE לסכמה הScientist שקיבלנו מאבינועם ואליעזר נצטרך מהם הרשאת reference:

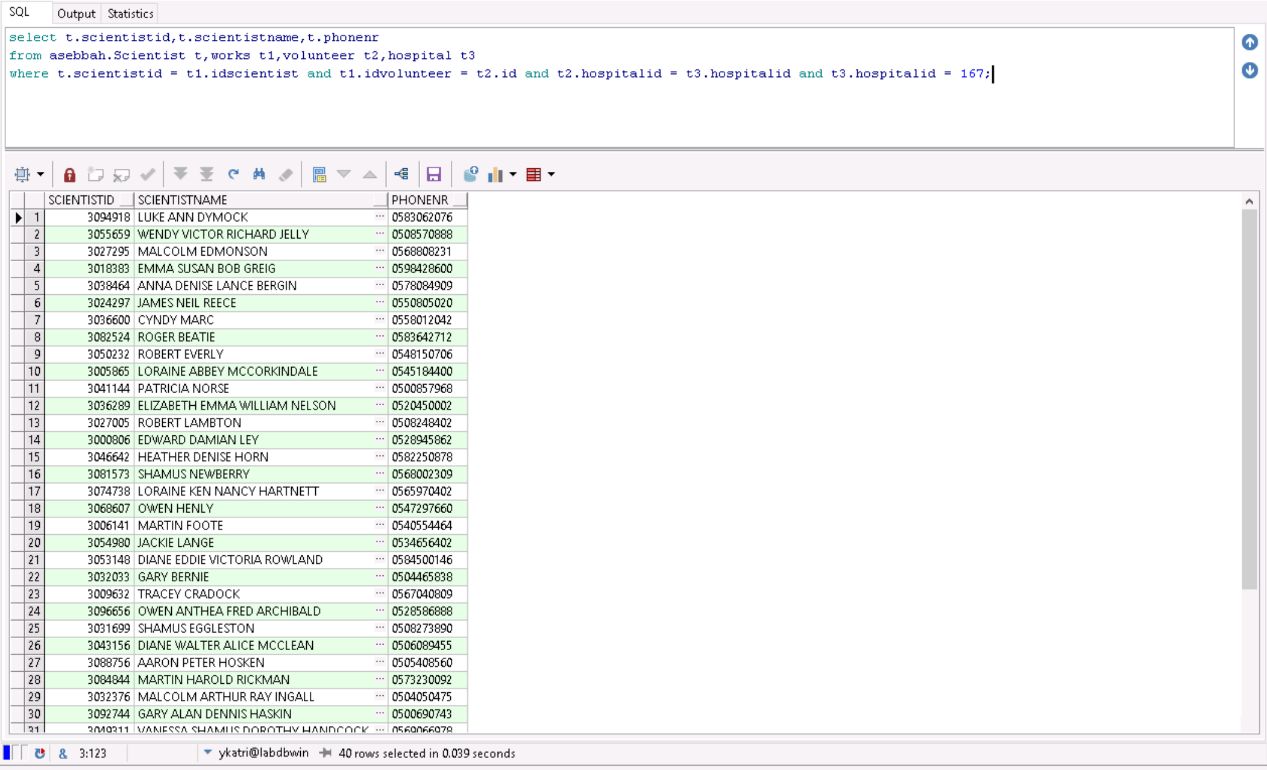


על מנת שנוכל לאכלס את הסכמה ע"י הסכמה שקיבלנו מאבינועם ואליעזר נצטרך מהם הרשאת select:

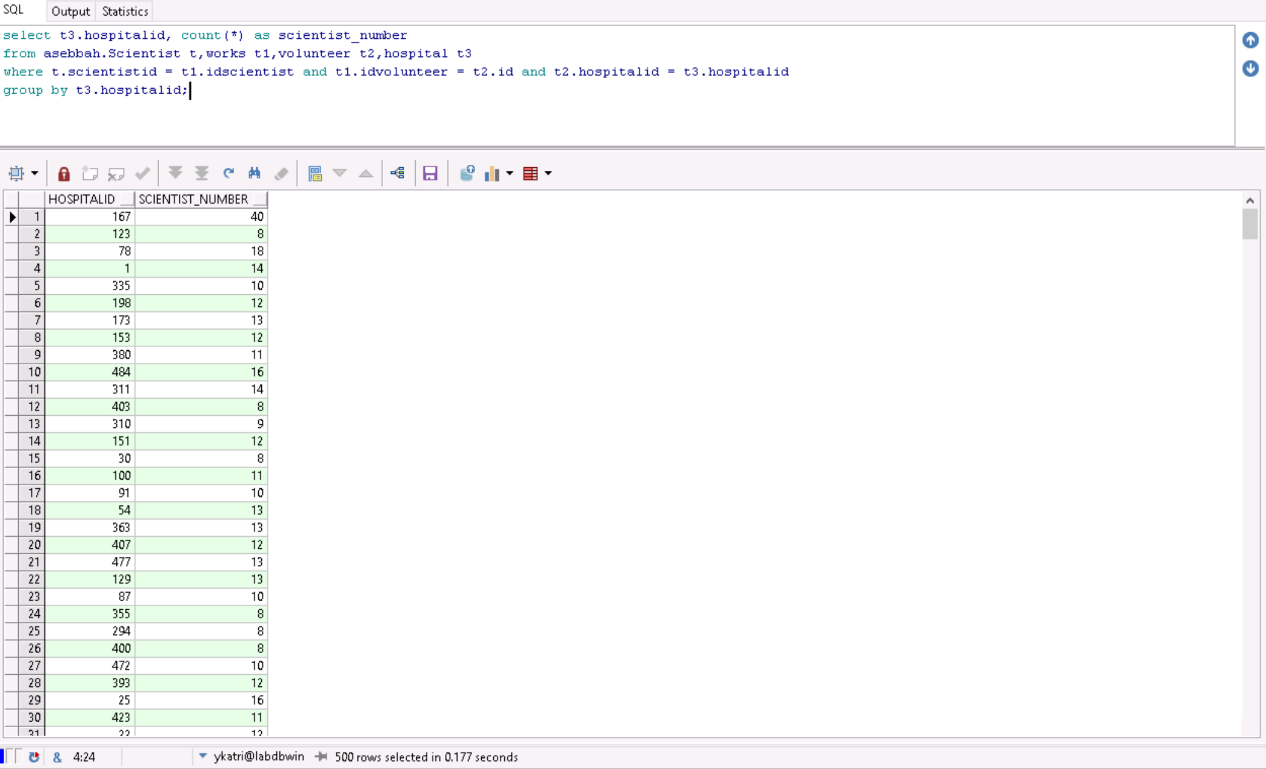


## שאילתות האינטגרציה

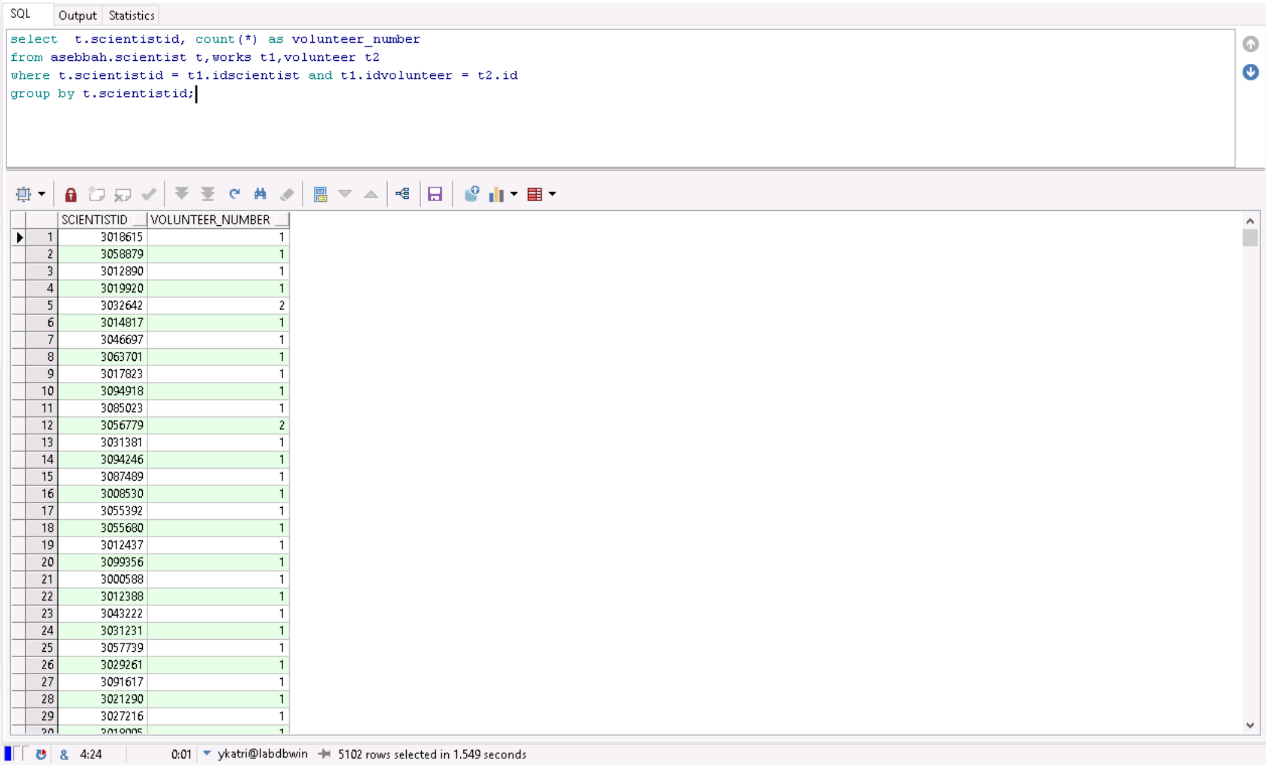
1. על מנת להקל על מציאת מדען נחוץ במקרה שתתעורר בעיה בקרבת בית חולים מסוים ניצור שאילתא שתחזיר את כל המדענים שנמצאים בבית חולים זה.



1. על מנת שנוכל לפזר תקציבים למחקר באופן צודק בין בתי חולים ניצור שאילתא שתחזיר לנו את כמות המדענים בכל בית חולים ונחלק את תקציב המחקר לפיה.

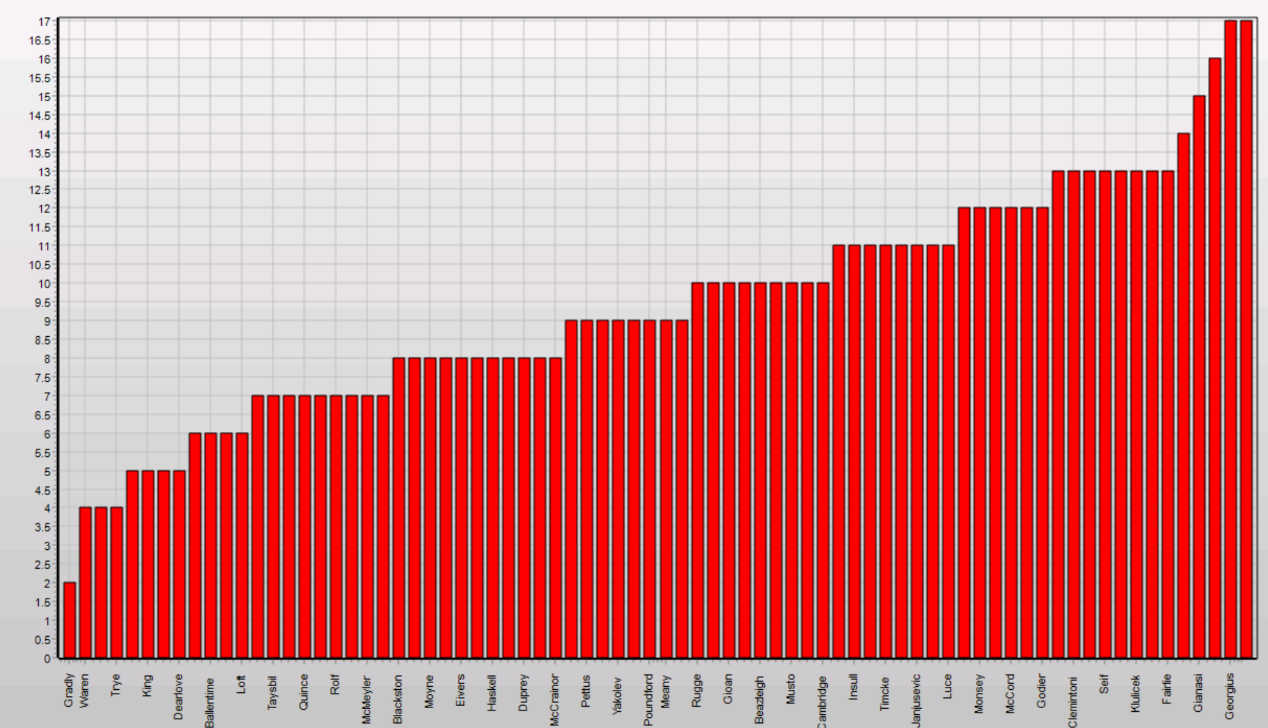


1. על מנת לפקח על מספר המתנדבים שיש לכל מדען ולחלק אותם באופן הגיוני ניצור שאילתא שתחזיר לנו את כמות המתנדבים שיש לכל מדען.

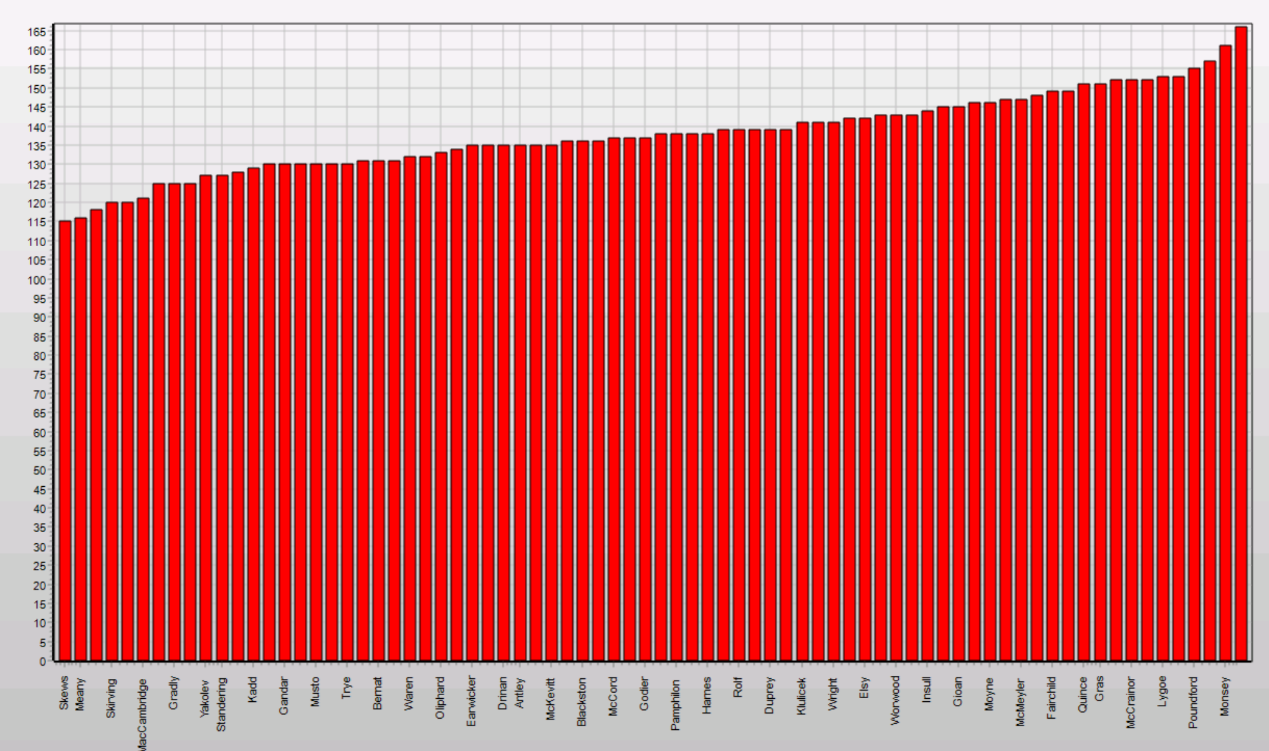


## תרשימים

1. תרשים המראה כמה מתנדבים דוקטורים יש בכל בית חולים שהתפוסה שלו מעל ל- 2700:



1. תרשים המראה כמה אמבולנסים יש בכל בית חולים שהתפוסה שלו מעל ל- 2700:



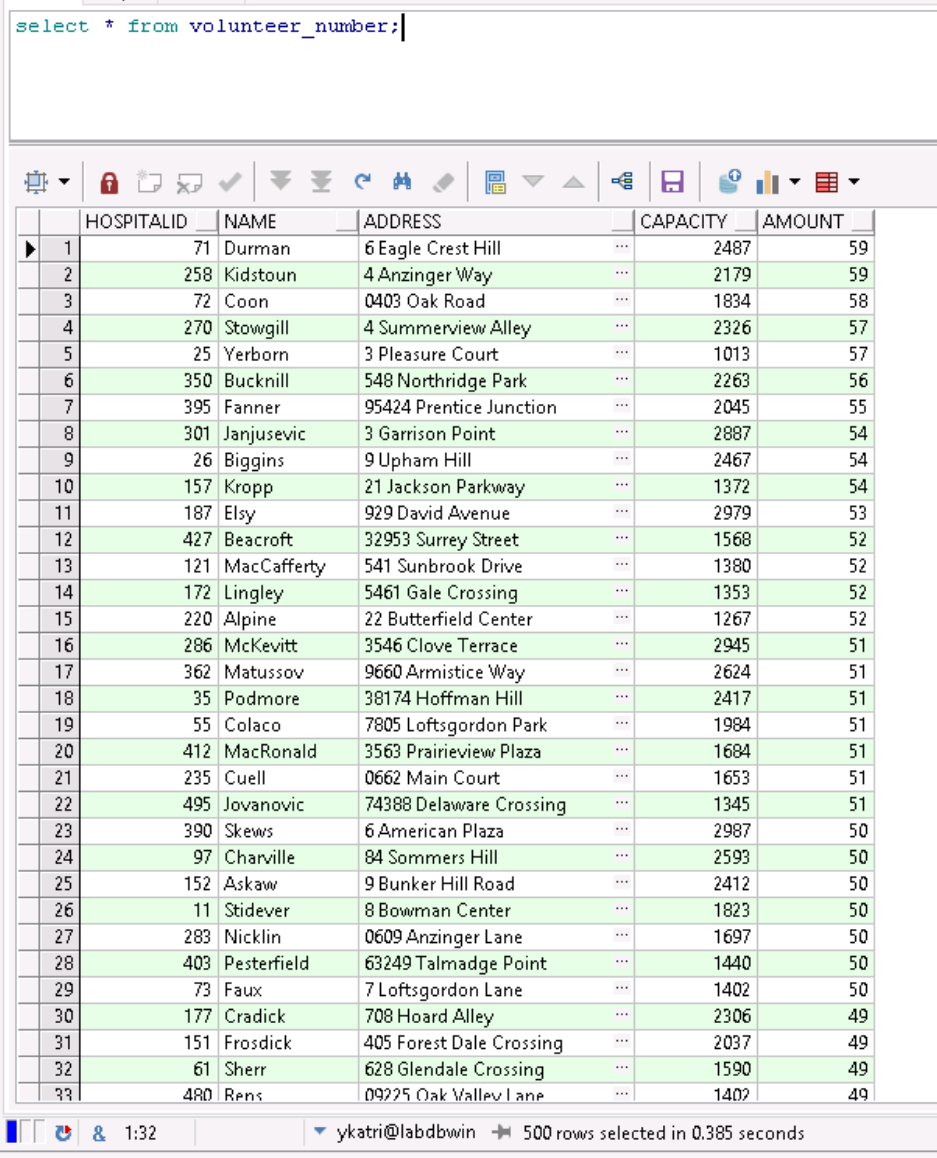
## Views

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.

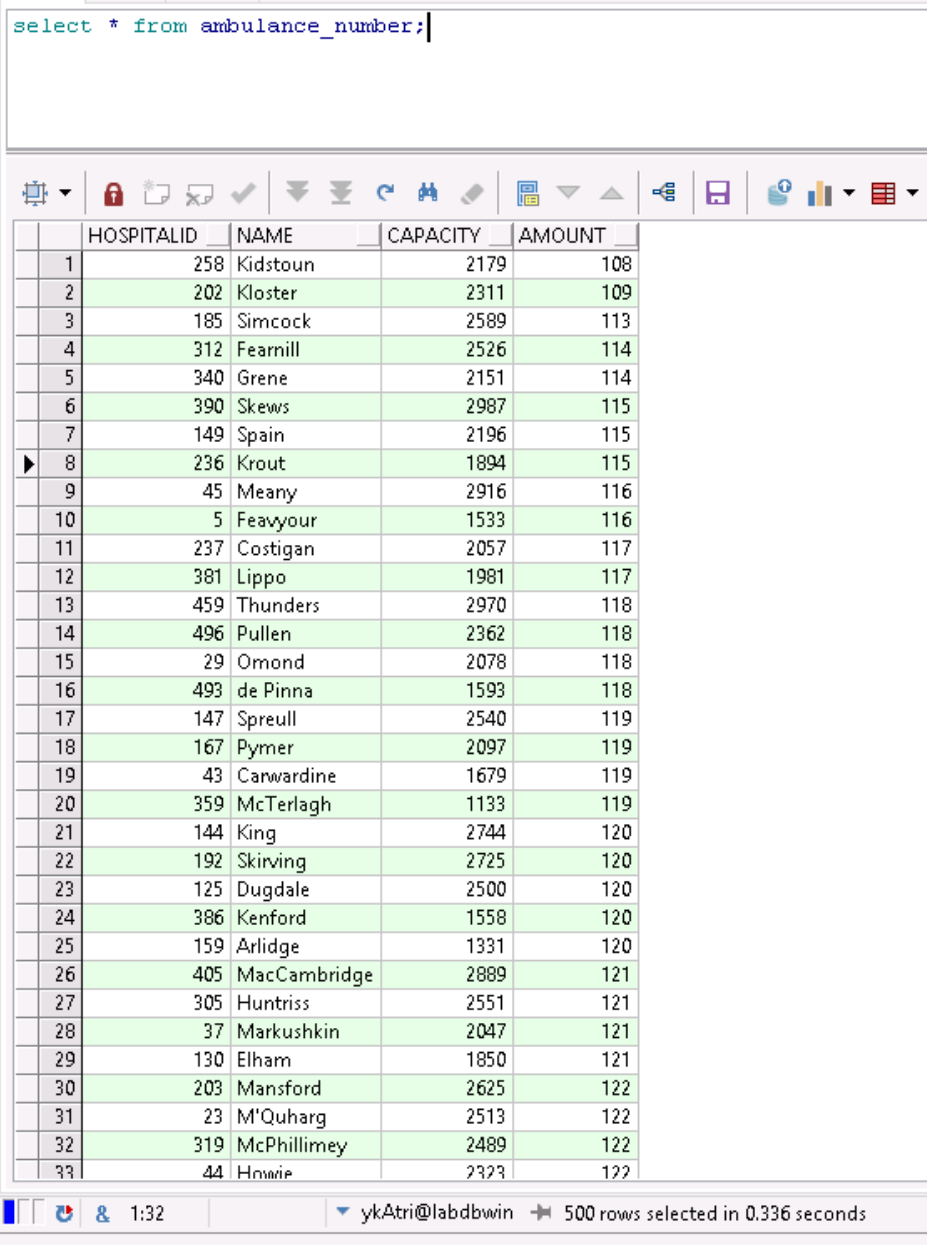
התייחסנו לשני משתמשים: הממשלה שמפקחת על מצב בתי החולים ובתי החולים שמטפלים בחולים.

1. הממשלה:

* כל פעם שקורה אסון כלשהו הממשלה מעוניינת לדעת לאיזה בית חולים לפנות על מנת לעזור לנפגעים ובעקבות כך היא תיגש הרבה פעמים לרשימת כמות המתנדבים בבתי החולים על מנת לדעת איזה בית חולים מתאים לה מבחינת כמות המתנדבים קיבולת ומיקום, לכן יצרנו view שמחזיר את רשימת בתי החולים הקיבולת ומ"ס המתנדבים שלהם והכתובת שלהם.

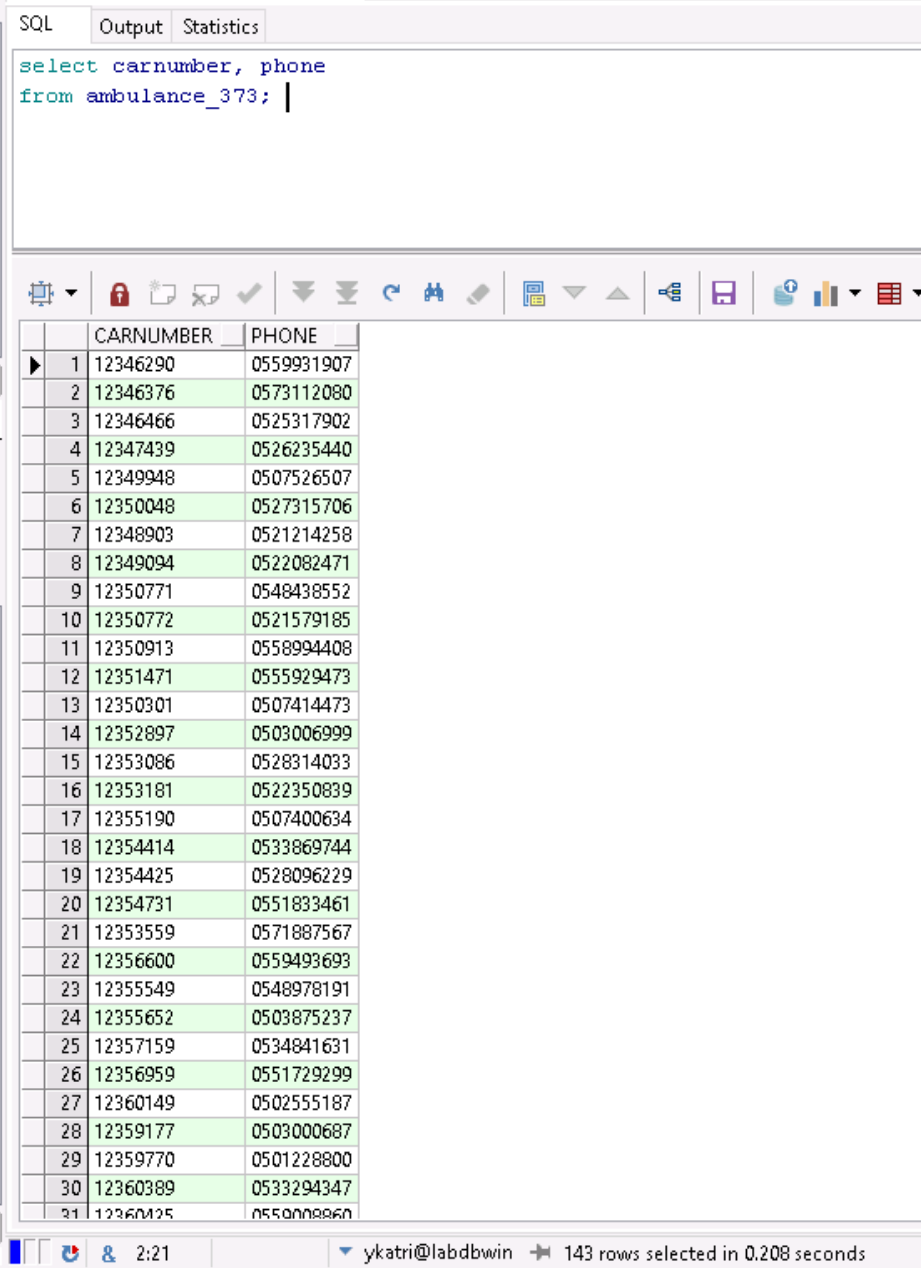


* כאשר הממשלה מעוניינת לחלק אמבולנסים לבתי החולים, כדי לתגבר אותם, היא תרצה להחליט לאילו בתי חולים לחלק ע"פ כמות האמבולנסים שיש להם והקיבולת שלהם ולכן יצרנו view שמחזיר את רשימת בתי החולים הקיבולת ומ"ס האמבולנסים שלהם.

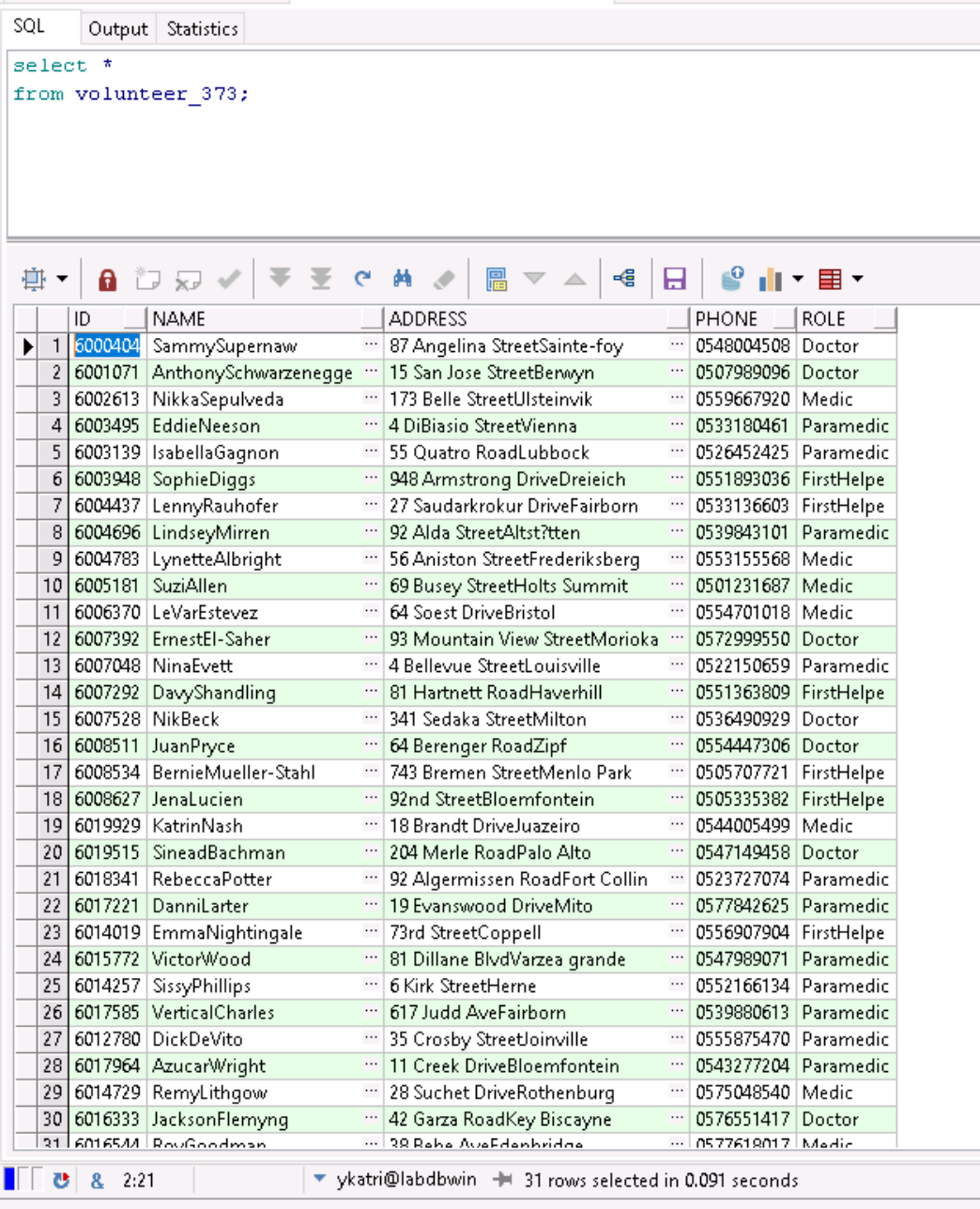


1. בית חולים:

* בכל מקרה חירום המוקדניות מבית החולים צריכות להתקשר לאמבולנסים הפנויים על מנת לשלוח אותם למקום האירוע או ליצור קשר עם אמבולנס שנמצא באירוע, בעקבות כך בית החולים ניגש למספרי הטלפון של האמבולנסים פעמים רבות מאוד ולכן יצרנו view והחזרנו את רשימת האמבולנסים של בית חולים ספיציפי ומספרי הטלפונים שלהם.

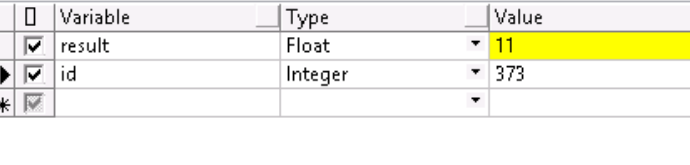
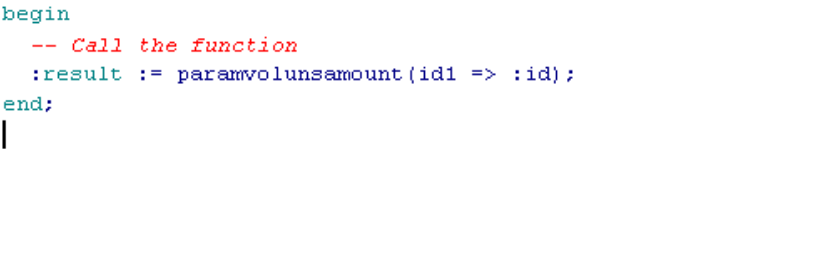


* כל פעם שצריך להזניק מתנדבים כדי לטפל באירוע מסוים, בית החולים צריך את רשימת המתנדבים שלו, בעקבות כך בית החולים ניגש למספרי הטלפון של המתנדבים פעמים רבות ולכן יצרנו view שמחזיר את רשימת המתנדבים של בית חולים ספיציפי.

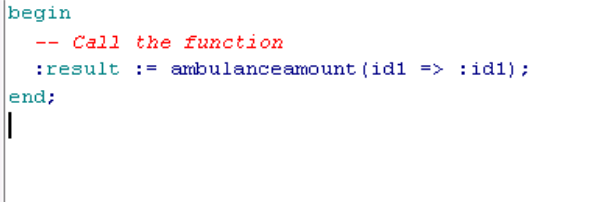
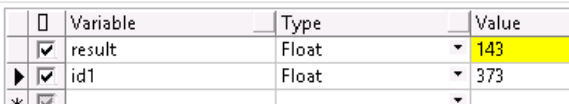


## פונקציות

* ParamVolunsAmount – הפונקציה מקבלת מספר זיהוי של בית חולים(integer) ומחזירה את כמות המתנדבים הפראמידיקים שיש שם ((integer. אם הקלט שגוי או שאין מתנדבים בבית חולים, הפונקציה תחזיר 0. הפונקציה הזאת מועילה לבתי החולים שרוצים לדעת כמה מתנדבים פארמדיקים יש אצלם.



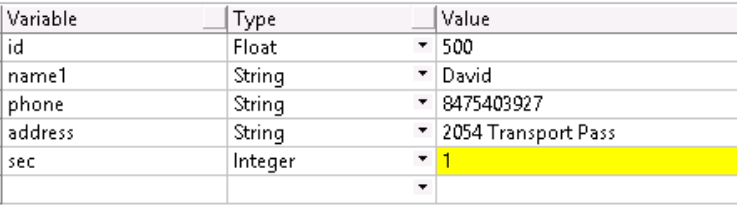
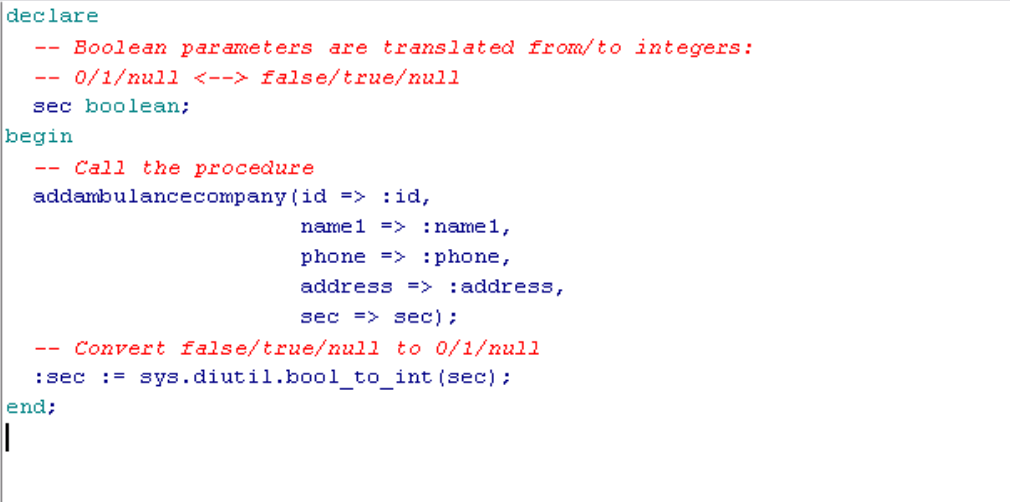
* ambulanceAmount – הפונקציה מקבלת מספר זיהוי של בית חולים(integer) ומחזירה את כמות האמבולנסים שיש שם ((integer. אם הקלט שגוי או שאין אמבולנסים לבית החולים, הפונקציה תחזיר 0. הפונקציה הזאת מועילה לבתי החולים שרוצים לדעת כמה אמבולנסים יש אצלם.

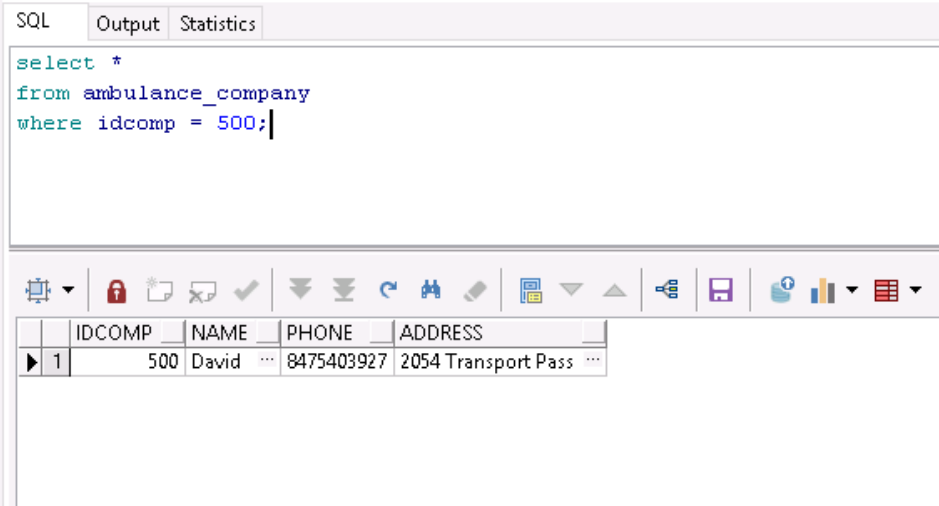


## פרוצדורות

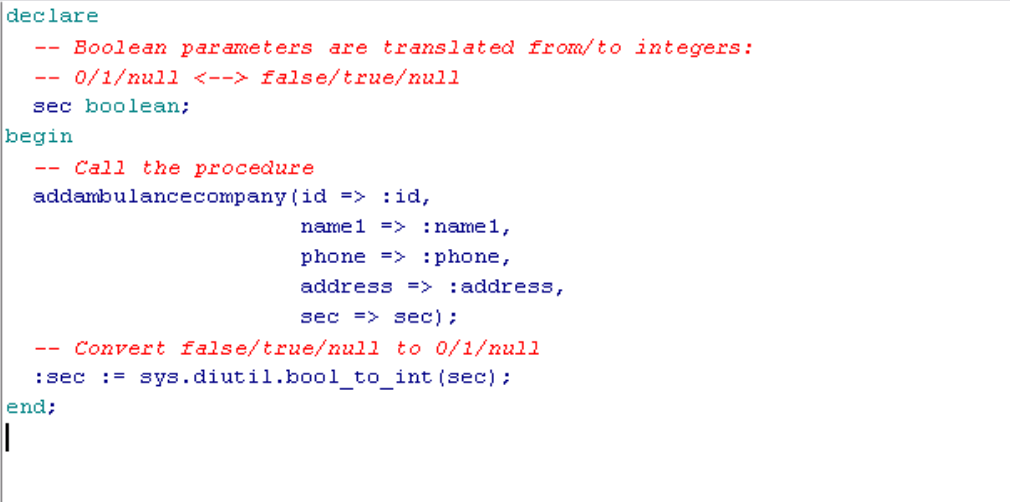
יצרנו 2 פרוצדורות שונות שמשקפות שני שימושים של פרוצדורה:

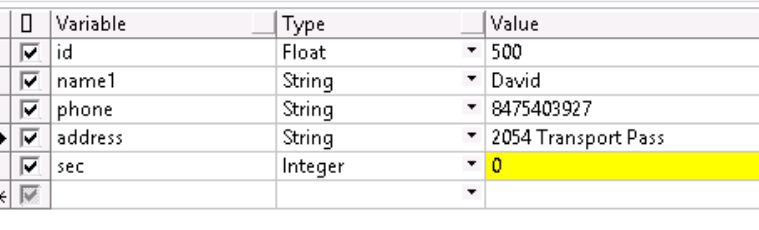
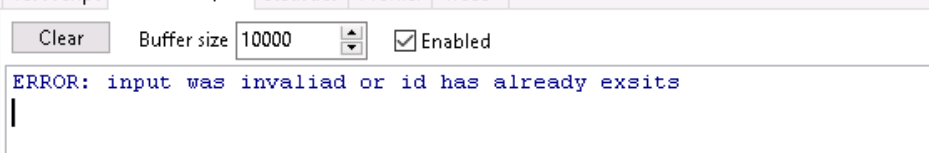
* addAmbulanceCompany- הפרוצדורה מוסיפה חברה חדשה(מ"ס זיהוי, שם, מ"ס פלאפון, כתובת) לתוך הסכמה של חברות האמבולנסים. אם היא מצליחה להוסיף לתוך הסכמה, היא מכניסה לתוך משתנה בוליאני sec אמת, אם היא לא מצליחה(לדוג': קיימת כבר חברה כזאת בסכמה עם אותו מ"ס זיהוי) היא מכניסה false ומדפיסה הודעת שגיאה למסך ה - DBMS output:
  + מצב הצלחה:



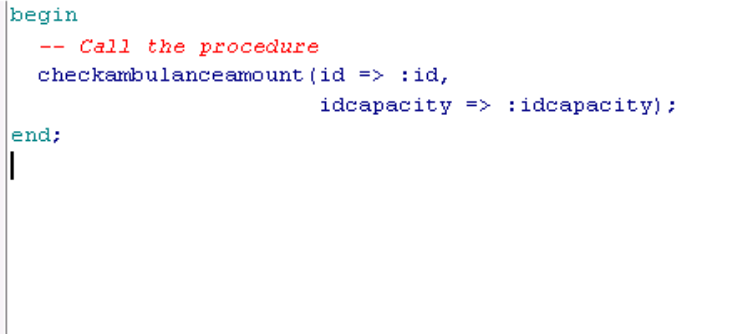
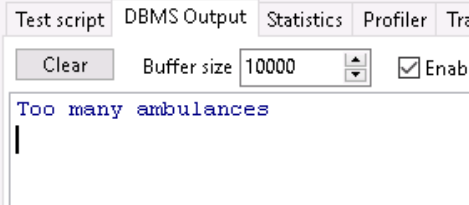
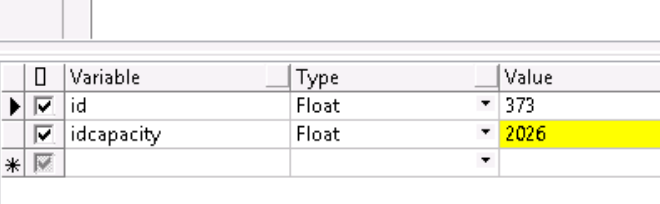


* + מצב תקלה(החברה כבר קיימת במאגר):





* CheckAmbulanceAmount – הפרוצדורה מקבלת כקלט מספר זיהוי של בית חולים(integer) ומחזירה דרך הערך idCapacity את הקיבולת של הבית חולים ומדפיסה האם יש מספיק אמבולנסים לבית חולים. בפרוצדורה הזאת אנחנו עושים שימוש בפונקציה ambulanceAmount .הפרוצדורה מועילה לבתי החולים שרוצים לדעת אם יש להם מספיק אמבולנסים, דוגמת הרצה של בית החולים שמספר הזיהוי שלו הוא 373:



# נספחים

## נספח ראשון: יצירת הטבלאות

1. CREATE TABLE ambulance\_company

(

IDComp NUMERIC(3) NOT NULL,

Name VARCHAR(20) NOT NULL,

Phone VARCHAR(10) NOT NULL,

Address VARCHAR(30) NOT NULL,

PRIMARY KEY (IDComp)

);

1. CREATE TABLE Hospital

(

Name VARCHAR(15) NOT NULL,

Address VARCHAR(30) NOT NULL,

HospitalID NUMERIC(3) NOT NULL,

Capacity NUMERIC(5) NOT NULL,

PRIMARY KEY (HospitalID)

);

1. CREATE TABLE volunteer

(

ID NUMERIC(9) NOT NULL,

Name VARCHAR(20) NOT NULL,

Address VARCHAR(30) NOT NULL,

Phone VARCHAR(10) NOT NULL,

role VARCHAR(10) NOT NULL,

HospitalID NUMERIC(3) NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (HospitalID) REFERENCES Hospital(HospitalID)

);

1. CREATE TABLE ambulance

(

Phone VARCHAR(10) NOT NULL,

CarNumber VARCHAR(8) NOT NULL,

IDComp NUMERIC(3) NOT NULL,

HospitalID NUMERIC(3) NOT NULL,

PRIMARY KEY (CarNumber),

FOREIGN KEY (IDComp) REFERENCES ambulance\_company(IDComp),

FOREIGN KEY (HospitalID) REFERENCES Hospital(HospitalID)

);

## נספח שני: שאילתות ואינדקסים

(האינדקסים מודגשים)

1. select ID ,Name

from VOLUNTEER t

where t.role = 'Doctor';

1. select t.hospitalid, t.name

from Hospital t

where t.capacity >=2000;

1. select t.hospitalid, t.capacity, t.address

from hospital t

order by capacity;

1. select t.idcomp, t.phone, t.address

from ambulance\_company t

order by idcomp;

1. select idcomp, t. name

from ambulance join ambulance\_company t using(idcomp)

where hospitalid = 373;

1. select t.carnumber, t.phone as car\_phone,idcomp, t1.phone as company\_phone, t1.address as company\_address

from ambulance t join ambulance\_company t1 using(idcomp)

where t.carnumber like '1234%'

order by t.carnumber;

1. select hospitalid,t.name,t.capacity,count(Distinct t1.id) as volunteers\_number, count(Distinct t2.carnumber) as ambulances\_number

from hospital t join volunteer t1 using(hospitalid) join ambulance t2 using(hospitalid)

group by hospitalid,t.name,t.capacity;

1. select t1.name,t1.idcomp,count(\*) as amount

from ambulance t, ambulance\_company t1

where t.idcomp = t1.idcomp

group by t1.idcomp,t1.name

order by t1.name;

1. **create index id\_index on ambulance(hospitalid);**
2. **drop index id\_index;**
3. **create index idcomp\_index on ambulance(idcomp);**
4. **drop index idcomp\_index;**
5. **create index role\_index on VOLUNTEER(role);**
6. **drop index role\_index;**

## נספח שלישי: Integration

**שאילתות grant:**

1. grant references on scientist to ykatri;
2. grant select on scientist to ykatri;

**יצירת טבלה:**

1. create table Works

(

IDVolunteer NUMERIC(9) Not null,

IDScientist INT not null,

primary key(IDVolunteer),

foreign key(IDVolunteer) references volunteer(id),

foreign key(IDScientist) references asebbah.Scientist(scientistId)

);

**שאילתות אינטגרציה:**

1. select t.scientistid, t.scientistname, t.phonenr

from asebbah.Scientist t,works t1,volunteer t2,hospital t3

where t.scientistid = t1.idscientist and t1.idvolunteer = t2.id and t2.hospitalid = t3.hospitalid and t3.hospitalid = 167;

1. select t3.hospitalid, count(\*) as scientist\_number

from works t1,volunteer t2,hospital t3

where t1.idvolunteer = t2.id and t2.hospitalid = t3.hospitalid

group by t3.hospitalid;

1. select t1.idscientist, count(\*) as volunteer\_number

from works t1,volunteer t2

where t1.idvolunteer = t2.id

group by t1.idscientist;

## נספח רביעי: views

1. create view volunteer\_number as

select hospitalid, t.name, t.address, t.capacity, count(\*) as amount

from volunteer join hospital t using(hospitalid)

group by hospitalid, t.name,t.capacity ,t.address

order by amount desc ,t.capacity desc;

2. create view ambulance\_number as

select hospitalid ,t.name, t.capacity, count(\*) as amount

from ambulance join hospital t using(hospitalid)

group by hospitalid, t.capacity, t.name

order by amount, t.capacity desc;

3. create view ambulance\_373 as

select t.carnumber, t.phone, t.idcomp

from ambulance t

where hospitalid = 373;

4. create view volunteer\_373 as

select t.id, t.name, t.address, t.phone ,t.role

from volunteer t

where hospitalid = 373;

## נספח חמישי: פונקציות ופרוצדורות

(הפרוצדורות מודגשות)

1. create or replace function ParamVolunsAmount(id1 in integer) return integer is

FunctionResult integer;

begin

select count(\*) into FunctionResult

from volunteer

where role = 'Paramedic' and hospitalid = id1;

return(FunctionResult);

end ParamVolunsAmount;

/

2. create or replace function ambulanceAmount(id1 in integer) return integer is

FunctionResult integer;

begin

select count(\*) into FunctionResult

from ambulance

where hospitalid = id1;

return(FunctionResult);

end ambulanceAmount;

/

3. **create or replace procedure addAmbulanceCompany(ID in integer, name1 in char, phone in char, address in char, sec out boolean) is**

**begin**

**insert into ambulance\_company values(ID,name1,phone,address);**

**sec := True;**

**Exception**

**when others then**

**dbms\_output.put\_line('ERROR: input was invaliad or id has already exsits');**

**sec := False;**

**end addAmbulanceCompany;**

**/**

4. **create or replace procedure CheckAmbulanceAmount(id in integer, IdCapacity out integer) is**

**begin**

**select capacity into IdCapacity**

**from hospital**

**where hospitalid = id;**

**if ambulanceamount(id1 => id) \*100 = IdCapacity then**

**dbms\_output.put\_line('the number of ambulance is ok');**

**else if ambulanceamount(id1 => id) \*100 > IdCapacity then**

**dbms\_output.put\_line('Too many ambulances');**

**else**

**dbms\_output.put\_line('lack of ambulance');**

**end if;**

**end if;**

**end CheckAmbulanceAmount;**

**/**