תוכן העניינים:

**חלק א:**

* תיאור המערכת והיחידה הנבחרת
* תרשים ERD
* תרשים DSD
* יצירת טבלאות
* הכנסת נתונים
* שאילתות

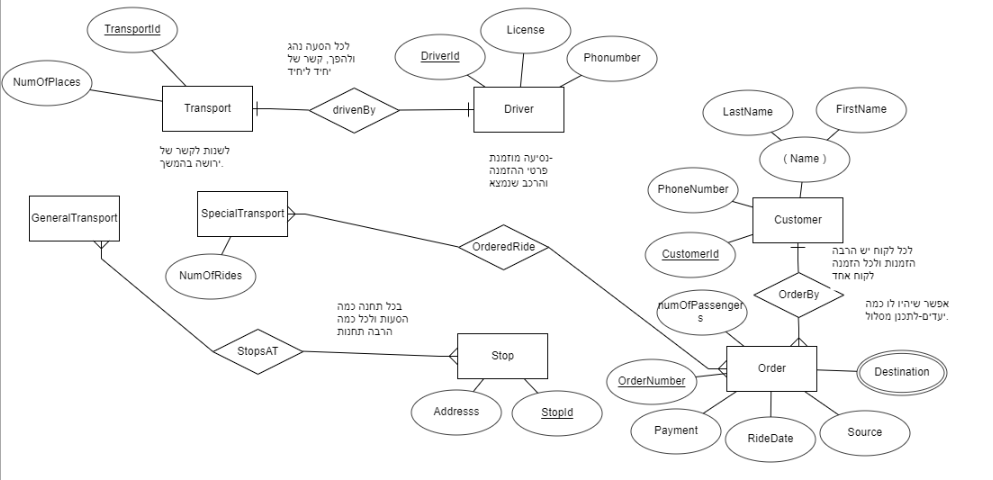
**חלק ב:**

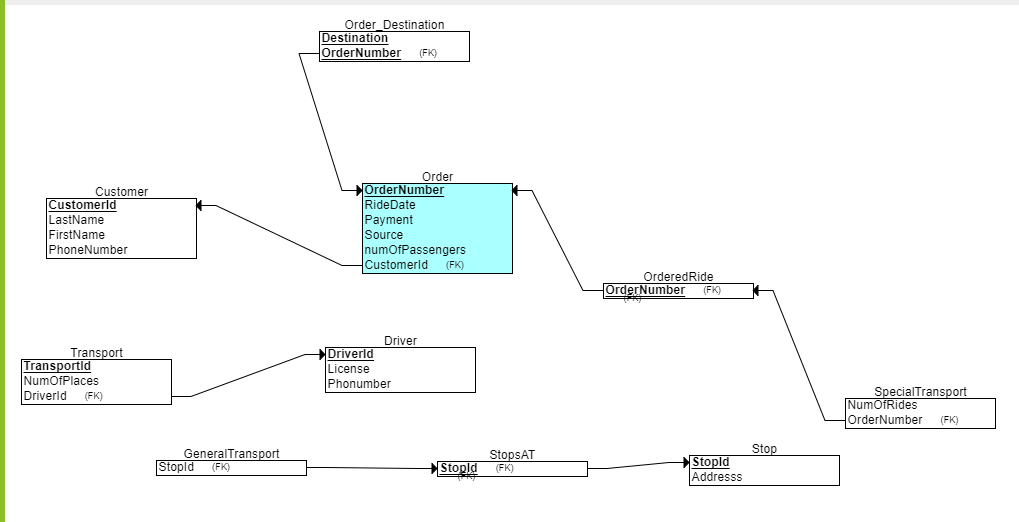
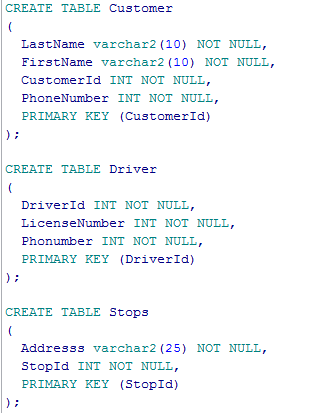
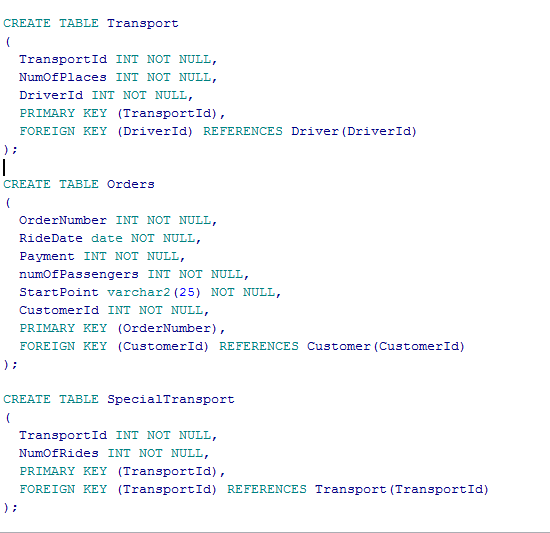
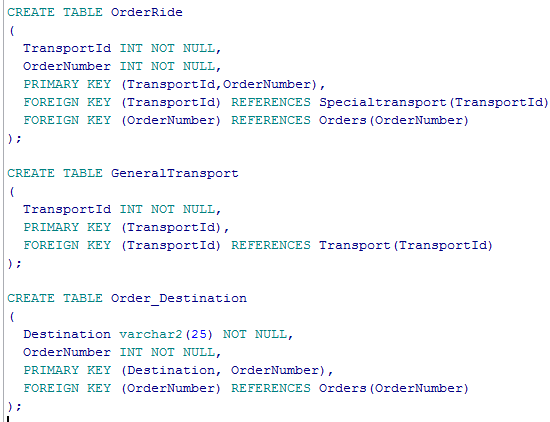
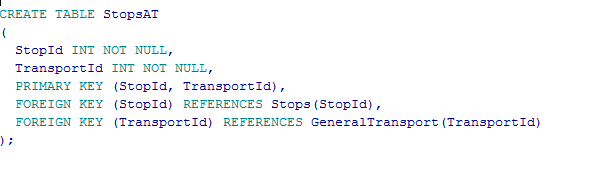
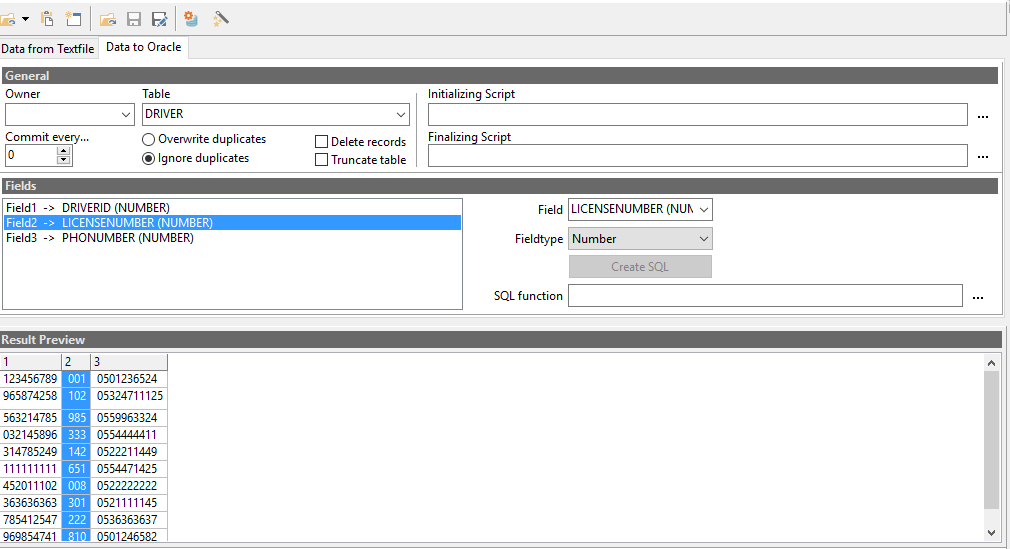
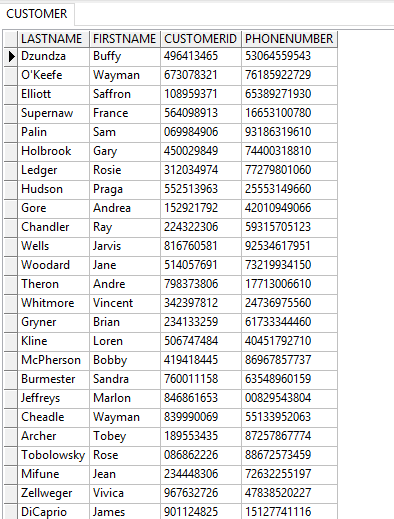
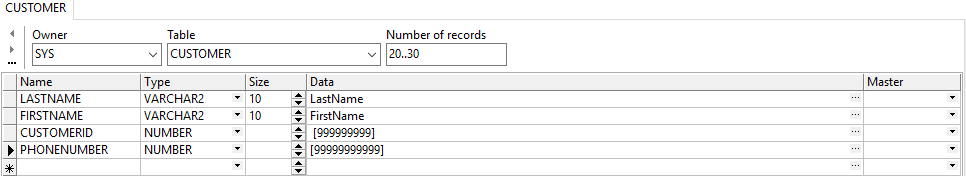
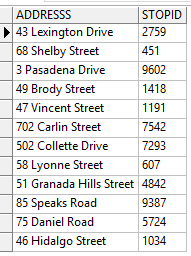
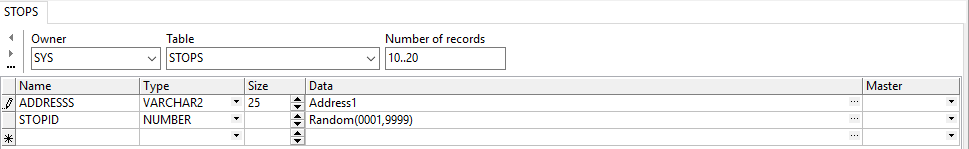
* תרשים INTEGRATED ERD
* אלגוריתם לשחזור הטבלאות
* -החלטות
* -התאמת השאילתות מחלק א לבסיס הנתונים המשולב
* אילוצים
* יצירת פרויקט
* מבטים
* Index
* GRANT&REVOKE

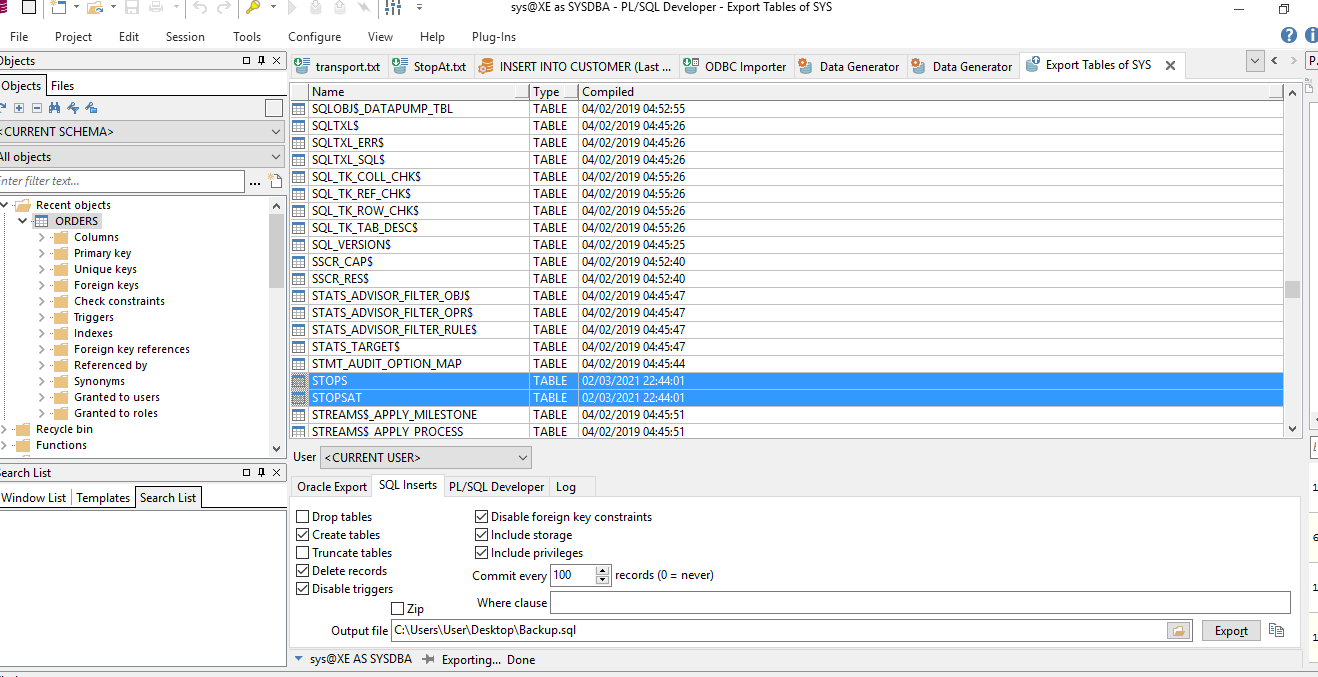
**חלק ג:**

* שאילתות עם פרמטרים
* פרוצדורות
* פונקציות
* טריגרים

תיאור המערכת והיחידה הנבחרת:  
המערכת הינה מערכת של חברת הסעות המספקת שירותי נסיעות קבועים וגם בהזמנות אישיות.   
**היחידה הנבחרת- קווים בהזמנה אישית ותיירות**.

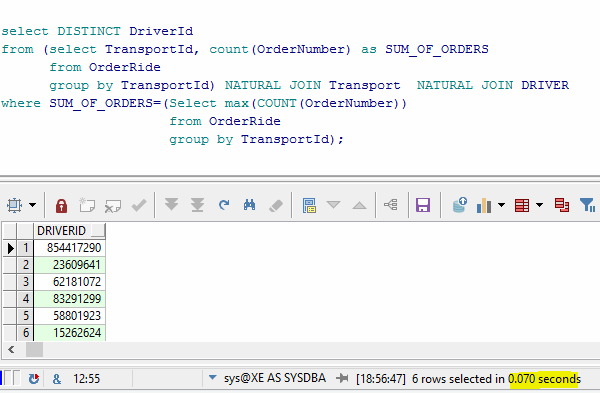
ניתן להזמין הסעה ממקור מסוים ליעד הרצוי בתאריך הרצוי בתחנות הרצויות ולמספר נוסעים לפי בקשת הלקוח, כמו כן ישנן נסיעות בעלות מסלול קבוע מראש המומלצות לתיירות.  
  
**ERD**

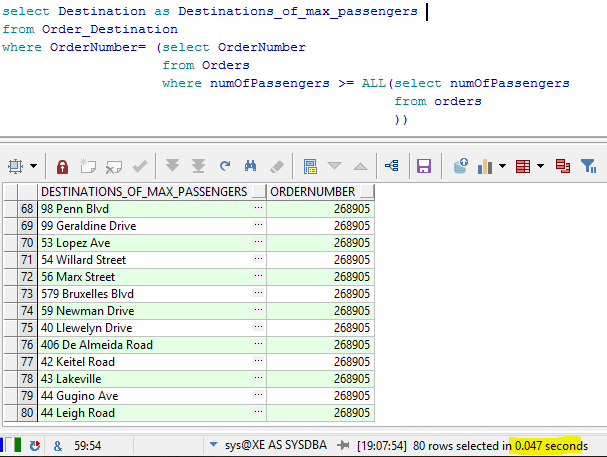
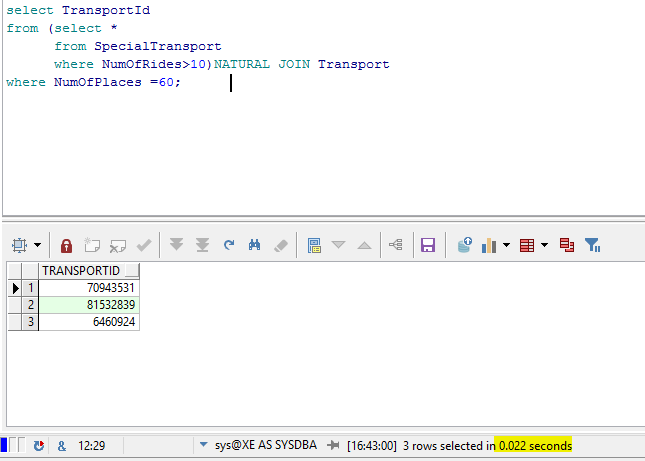
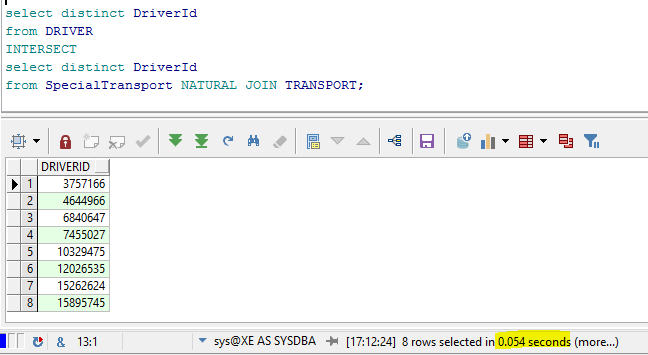
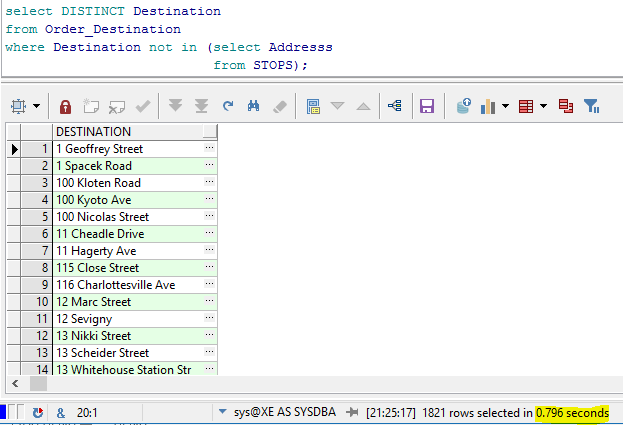
**DSD**  
  
  
**יצירת הטבלאות:**  
   
   
**הכנסת נתונים ע"י קובץ טקסט:**  
  
  
  
  
**הכנסת נתונים ע"י מחולל נתונים:**  
 **\***עשינו ל6 טבלאות, ניתן לראות זאת בקבצים המצורפים (אך לא צירפנו צילומי מסך של הכל). **גיבוי: sql Insert**



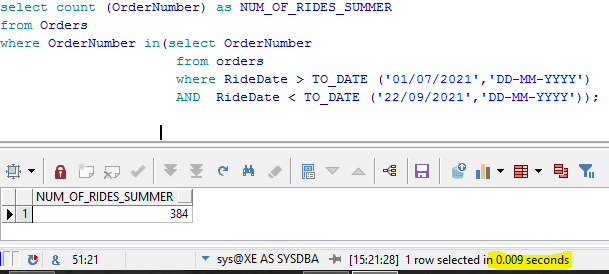
**שאילתות**

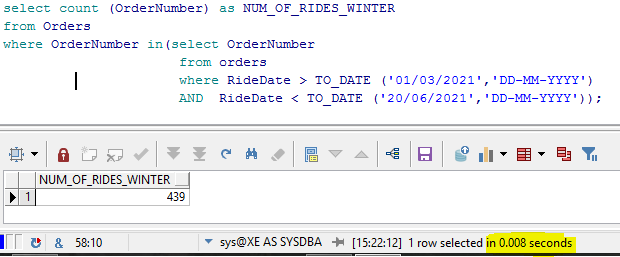
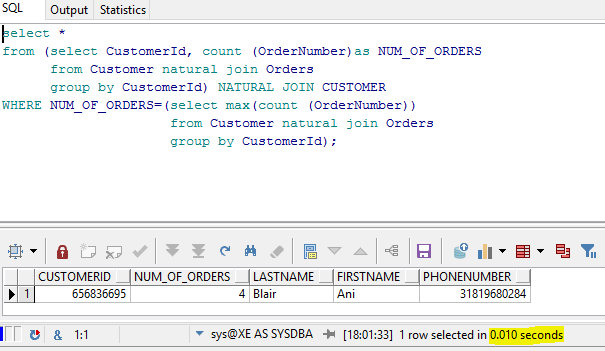
***Select***

1. שאילתה שמוצאת את הנהג שהרכב שלו הוזמן הכי הרבה פעמים.   
     
     
   2. שאילתה שמחזירה את היעדים של ההזמנות עם הכי הרבה נוסעים.

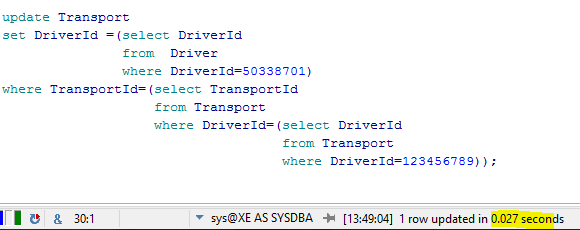
  
  
  
3. שאילתה מקוננת המחזירה את כל האוטובוסים המיוחדים בעלי 60 מקומות שהוזמנו לפחות 10 פעמים (לבדוק את הביקוש של האוטובוסים).  
  
  
  
4. שאילתה המחזירה את כל הנהגים של ההסעות המיוחדות.  
  
  
5. רשימת כל היעדים שהוזמנו ולא קיימת בהם תחנה רשמית.  
  
  
6. רצו לבדוק את מספר הנסיעות שהוזמנו בחודשי הקיץ לעומת כמות הנסיעות שהוזמנו בחודשי החורף.(2 שאילתות נפרדות אחת עבור מספר הנסיעות בקיץ ואחת עבור מספר הנסיעות בחורף).

מספר הנסיעות הכולל שהוזמנו בחודשי הקיץ.

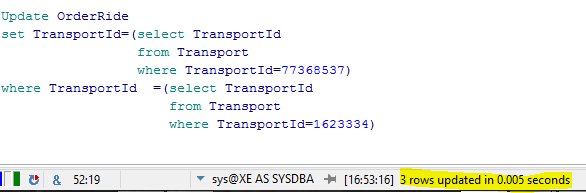
  
7.מספר הנסיעות הכולל שהוזמנו לחודשי החורף.

  
  
8. שאילתה שמחזירה את הלקוח שביצע הכי הרבה הזמנות מבין כל הלקוחות, ע"מ לתת לו תשופר.  
  
***Update***

1.נהג בעל ת.ז יצא לפנסיה ונהג אחר קיבלת את הרכב שלו ונדרש לעדכן זאת במערכת.

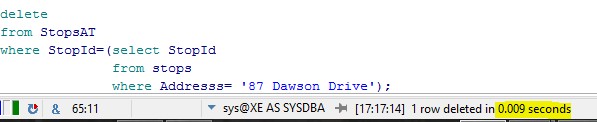


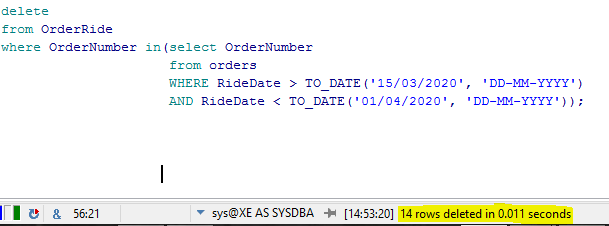
1. רכב בעל מספר xxxxישן מידי והפסיק לעבוד, רכב בעל מספר yyyy יקבל את הנסיעות שהוזמנו עבורו ועדיין לא התבצעו ויש לעדכן זאת במערכת.



***Delete***

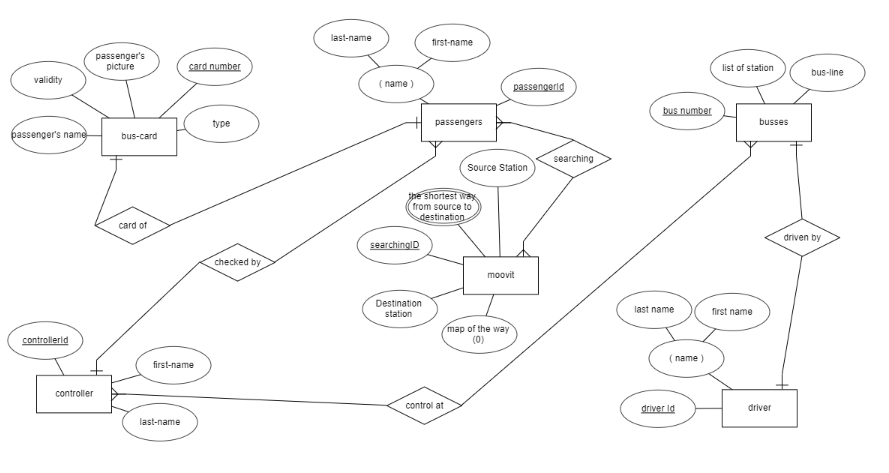
1. בשל עבודות בכביש נחסם המעבר דרך תחנה '87 Dawson Drive' ולכן נמחק את התחנה הזאת מכל המסלולים שעוברים דרכה.



1. בגלל הסגר בוטלו כל הנסיעות שנקבעו לתאריכים אלו ויש למחוק אותן מהמערכת.  
   

**חלק ב'- אינטגרציה:**

**ERD של היחידה הראשונה:**

 **integrated ERD:  
אלגוריתם לפיו פעלנו בשחזור הERD מהטבלאות.  
לקחנו את כל השדות שבטבלאות והוספתנו אותם כתכונות לישות בerd למשל:  
CREATE** **TABLE** buses

(

busLine INT **NOT** NULL,

list\_of\_stations VARCHAR2(30)**NOT** NULL,

bus\_number INT **NOT** NULL,

PRIMARY **KEY** (bus\_number)

); **אז busLine,list\_of\_stations,bus\_umber הם התכונות של הטבלה buses.  
בכל טבלה בדקנו מהן התכונות שהן primery key וסימנו אותם כמפתחות של הישות בסכמה ERD.**

**למשל לפי הטבלה למעלה מהשורה** PRIMARY **KEY** (bus\_number) **הבנו שbus\_number הוא המפתח בטבלה והוספנו אותו להיות המפתח של הישות buses בerd.  
כאשר בטבלה של ישות היה foreign key הבנו שזה קשר בין הטבלה הנוכחית לזו שבReference.  
בטבלה שהתכונות שהן המפתח הן גם ב foreign key אז זה קשר של רבים לרבים בין הטבלאות אליהן מתייחסים בReference. למשל:  
CREATE** **TABLE** controll\_at

(

controllerId INT **NOT** NULL,

bus\_number INT **NOT** NULL,

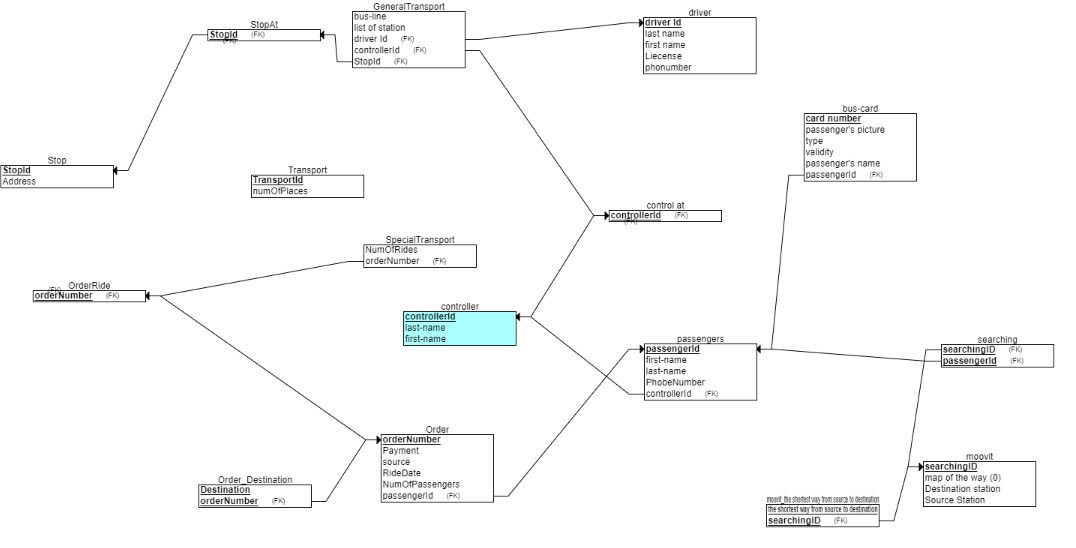
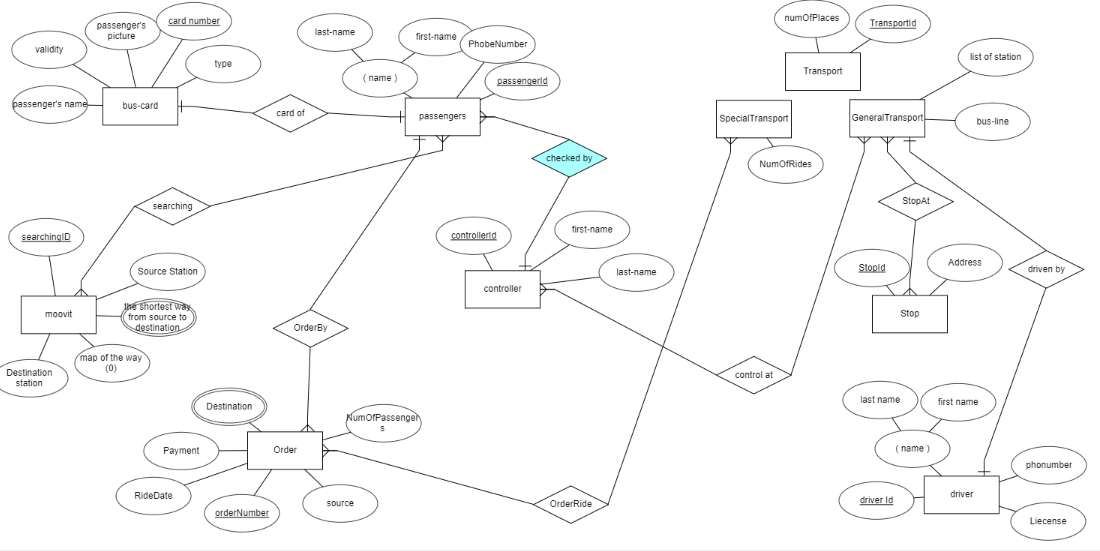
PRIMARY **KEY** (controllerId, bus\_number),

FOREIGN **KEY** (controllerId) **REFERENCES** controller(controllerId),

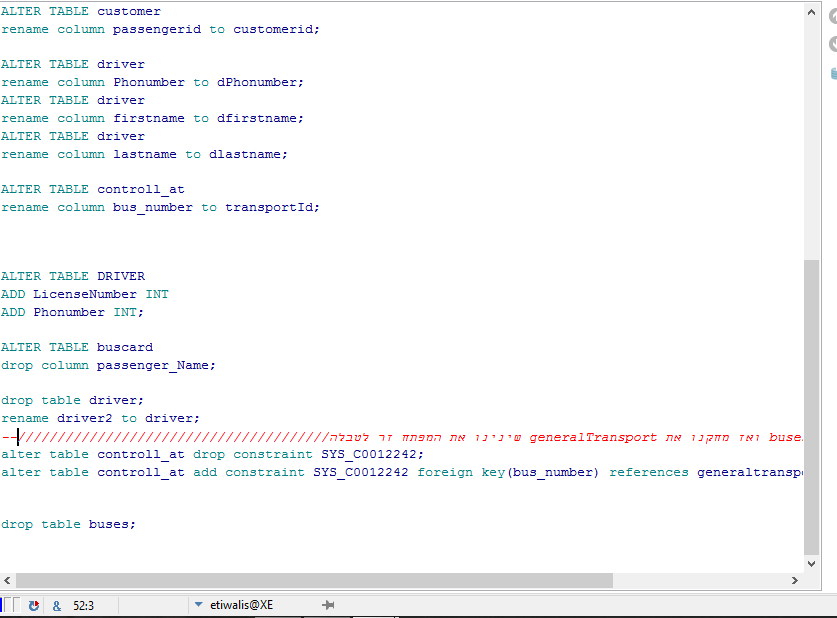
FOREIGN **KEY** (bus\_number) **REFERENCES** buses(bus\_number)

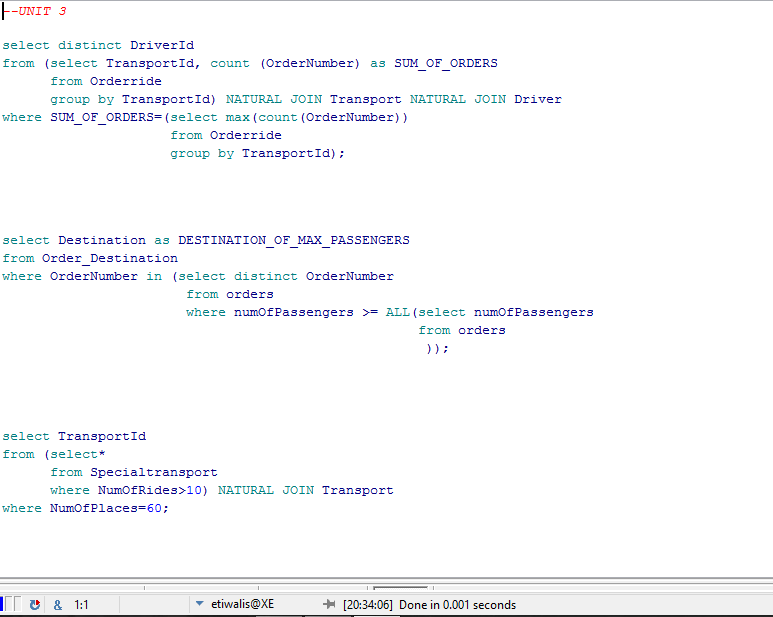
);

**ControIlAt זו טבלה שמציינת קשר של רבים לרבים בין 2 הישויות buses ו controller.  
  
   
החלטות:  
  
1. שילבנו את הישות Driver מ2 התרשימים לישות אחת עם כל התכונות.  
2. שינינו את הbus של היחידה הראשונה ל General Transport.  
3. הוספנו ל General Transportאת התכונהbusline ובמקום רשימה של תחנות יש לנו את הקשר stopsat.  
4. passengers וcustomer השארנו כ2 ישויות נפרדות כאשר passengers עבור נוסעים בתחבורה הציבורית ו customerעבור לקוחות שמזמינים נסיעות פרטיות.**

**5.מחקנו מהישות של busCard את התכונה: 'שם הנוסע', כיון שהיא לא תכונה של הכרטיס אלא באה לידי ביטוי בקשר.**

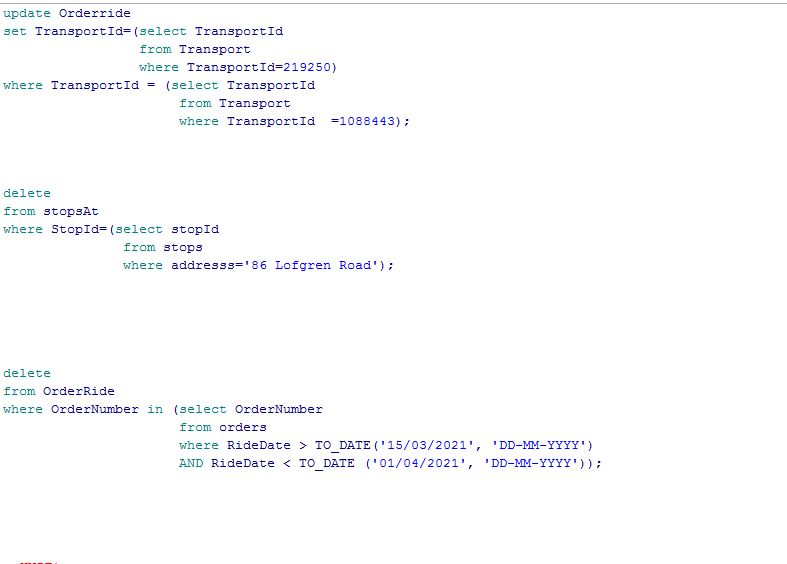
**Alter table:**

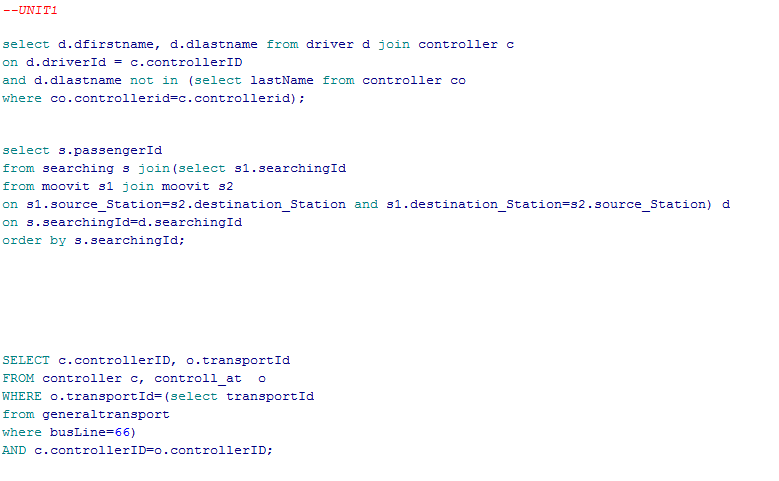


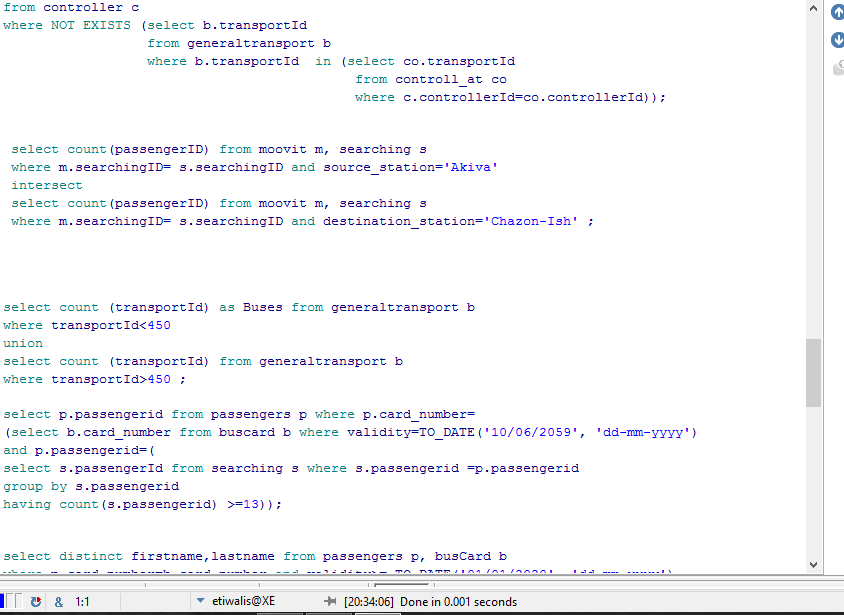
תיקון השאילתות שיתאימו לבסיס הנתונים החדש:  


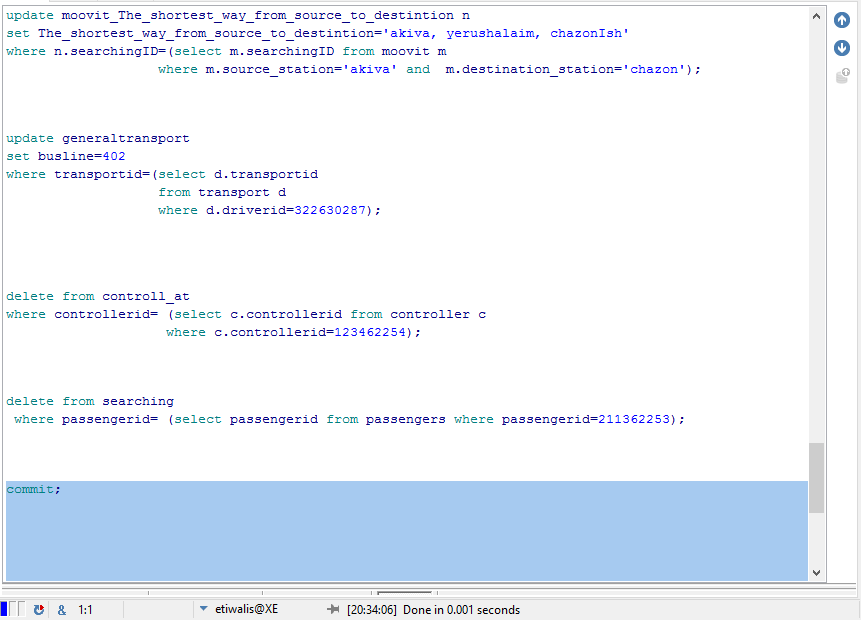




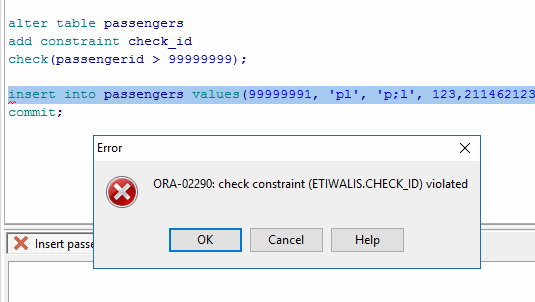


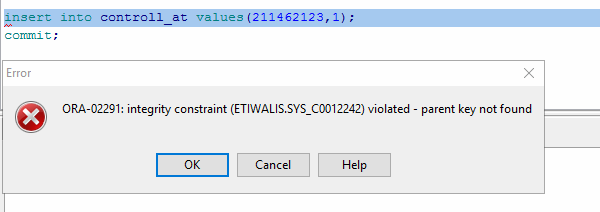




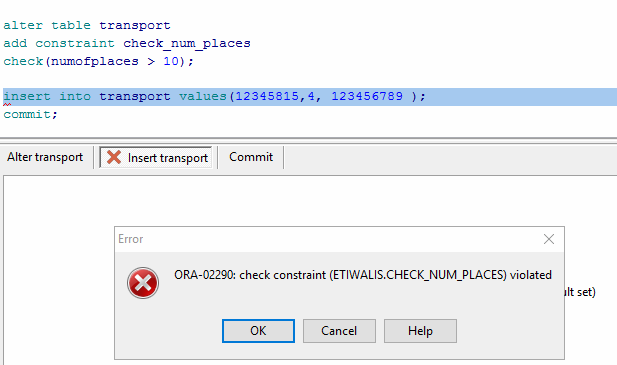


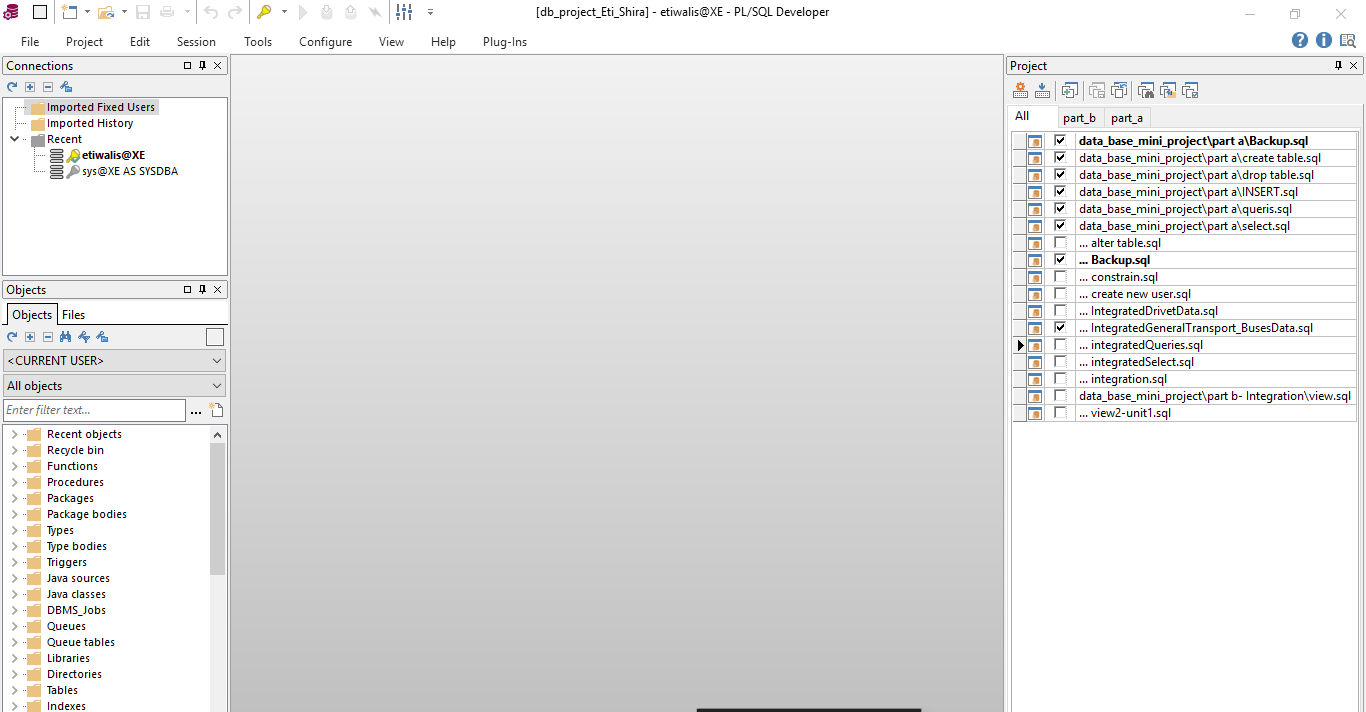
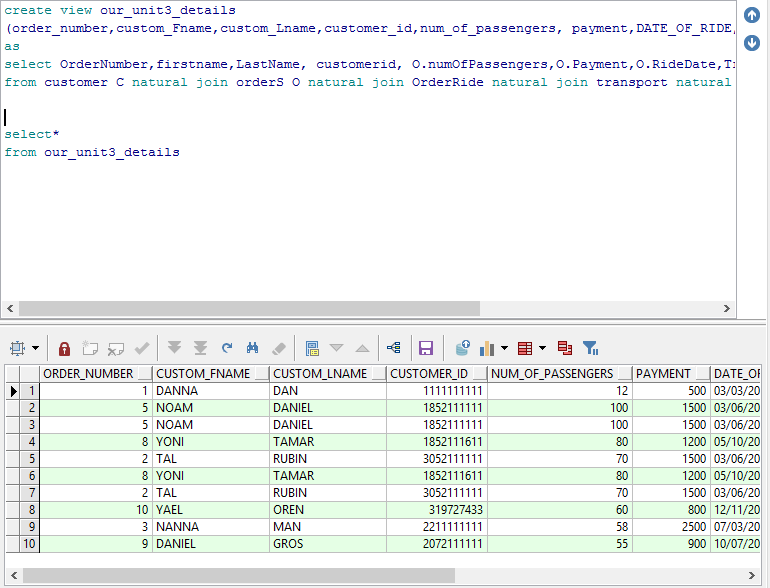
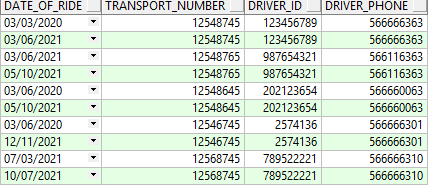
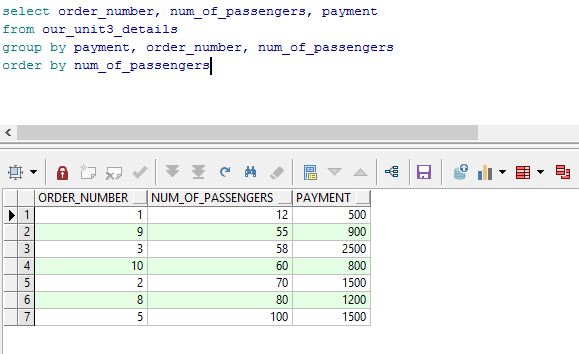
אילוצים:  
1.לבדוק שת.ז של נוסע בעלת 9 ספרות.

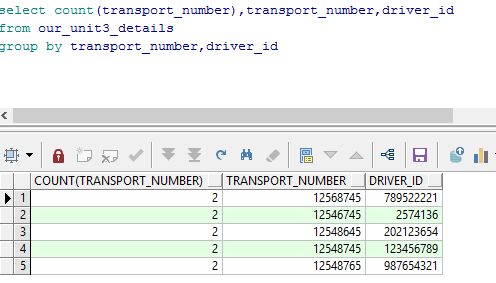
  
  
2. 

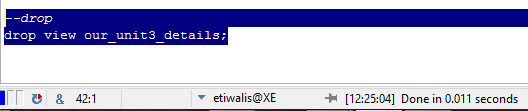


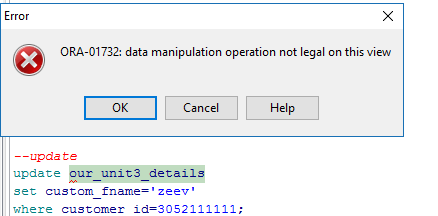
כשניסינו להוסיף שורה עם מספר אוטובוס לא קיים, קיבלנו את השגיאה הנ"ל.  
3.אילוץ שמספר המקומות המינימלי ברכב של החברה הוא 10.

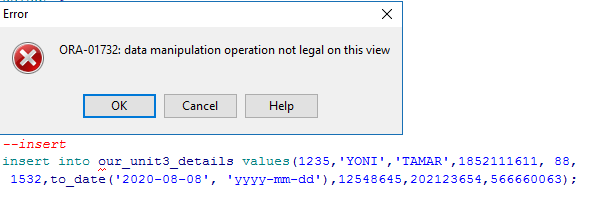
  
  
**יצירת פרויקט:**

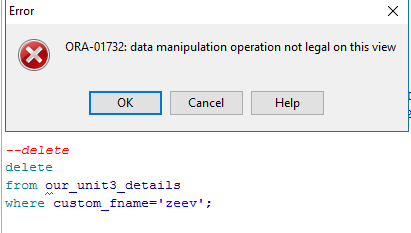
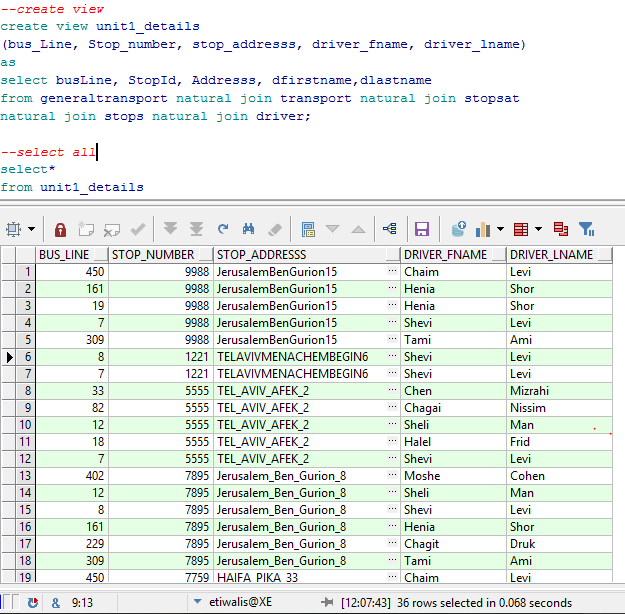
  
**view:**\*נרצה להציג טבלה (view) מסודרת עם כל פרטי הנסיעות המיוחדות.  
שאילתות על המבט:  
1. שאילתה שמחזירה את הנסיעות מקובצות לפי מספר ומסודרות מהמחיר הנמוך לגבוה.  


2.שאילתה שמחזירה את הרכב, הנהג וכמות הפעמים שהרכב הוזמן.  
  
drop-  
מצליח, שום view לא תלוי בו לכן אין בעיה לבטל את הview.

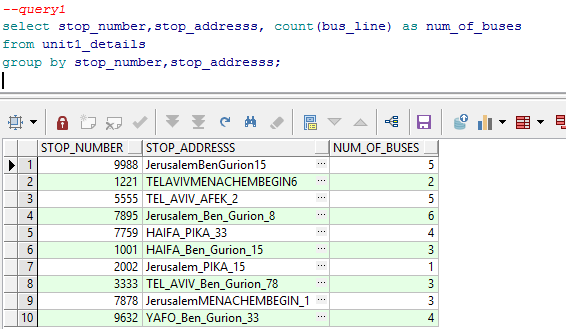


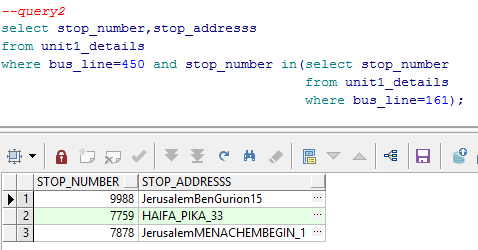
Update-   
 לא מצליח, סיבה: נעשה ניסיון להשתמש בUPDATE, INSERT או DELETE כאשר יש שימוש בdistinct ביצירת הview.   
  
insert-

כנ"ל.  
  
delete-

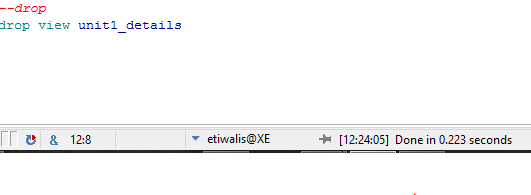
כנ"ל  
  
\*נרצה להציג טבלה עבור האוטובוסים, הנהגים והתחנות שלהם.  
  


שאילתות על המבט:  
1. שאילתה שמחזירה את כל התחנות עם כמות האטובוסים שעוצרים בכל תחנה.



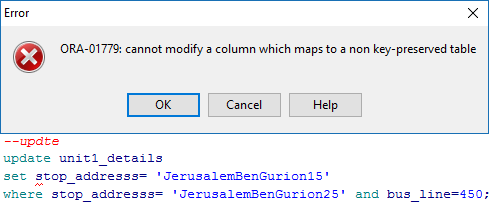
2.שאילתה שמחזירה את כל התחנות המשותפות לקו 450 ולקו 161.  


drop-  
מצליח, שום view לא תלוי בו לכן אין בעיה למחוק.

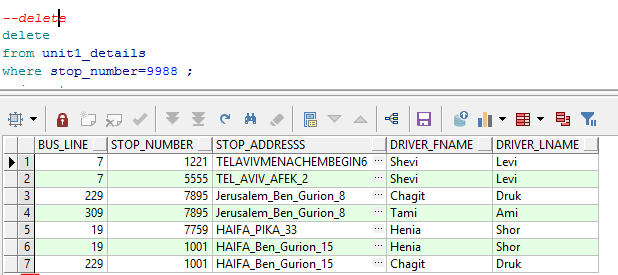


Update-

ננסה לעדכן תחנה מסויימת שזזה להמשך הרחוב.

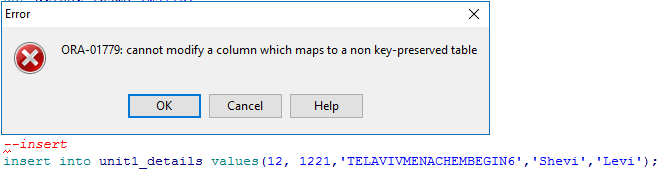
לא עובד-הוא לא מצליח לשייך אות השורה למפתח ראשי.  


Delete-  
המחיקה עבדה. נראה שאכן כל השורות עם התחנה 9988 נמחקו מהטבלה.



Insert-

לא עובד- אותה סיבה של העדכון.

  
  
**index:**  
index1:  
עבור address בטבלה stops.

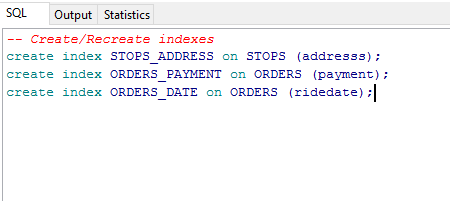
create index STOPS\_ADDRESS on STOPS (addresss);   
index2:

עבור payment שמכיל טווח גדול של ערכים.

create index ORDERS\_PAYMENT on ORDERS (payment);  
index3:

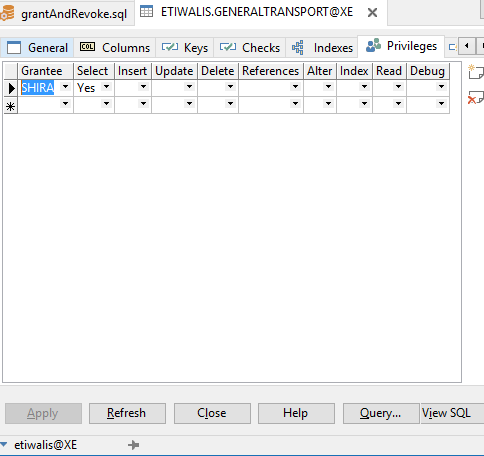
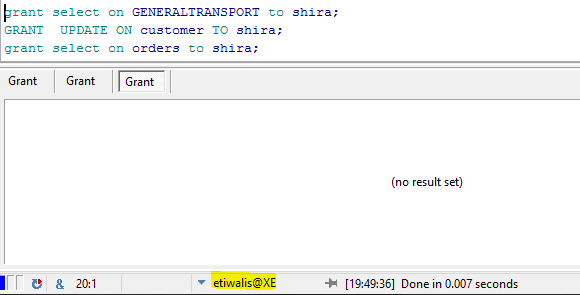
עבור ride\_date שמכיל טווח גדול של ערכים.

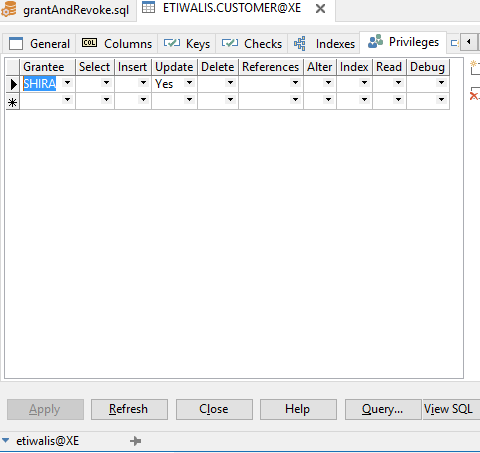
create index ORDERS\_DATE on ORDERS (ridedate);

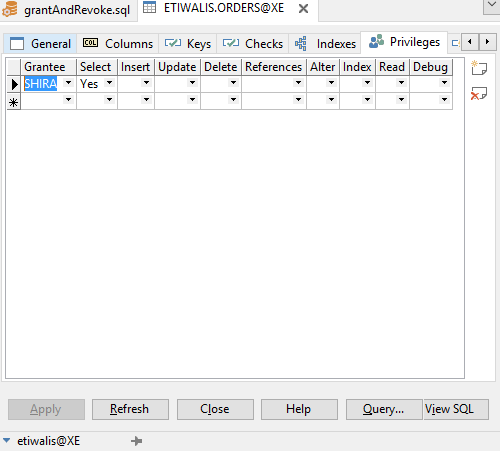
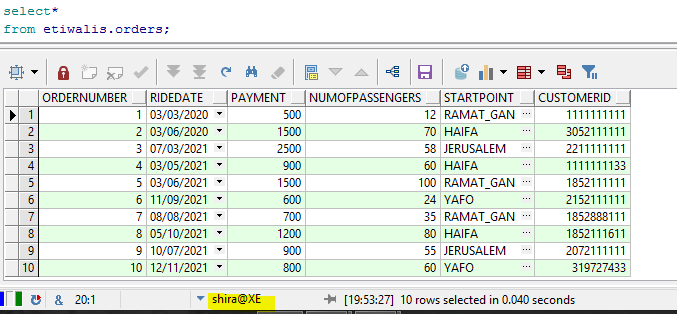
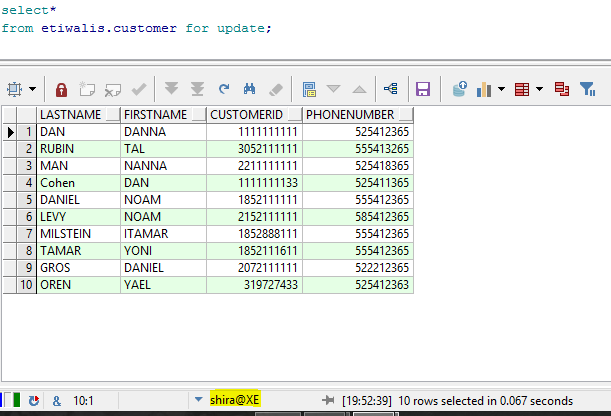
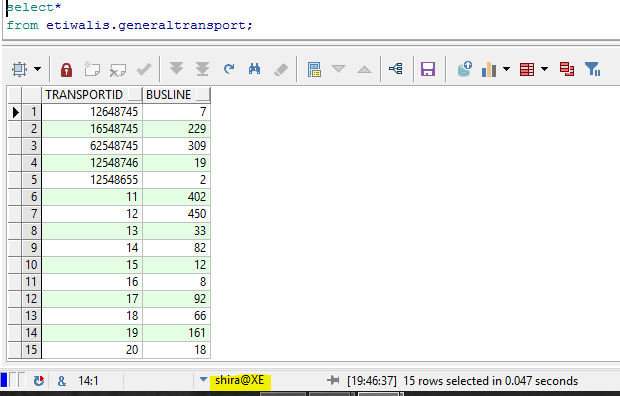


|  |  |
| --- | --- |
| לפני | אחרי |
| 0.07 | 0.011 |
| 0.047 | 0.017 |
| 0.232 | 0.218 |
| 0.054 | 0.046 |
| 0.796 | 0.033 |
| 0.009 | 0.007 |
| 0.008 | 0.008 |
| 0.010 | 0.008 |
| 0.027 | 0.001 |
| 0.005 | 0.001 |
| 0.009 | 0.002 |
| 0.011 | 0.001 |

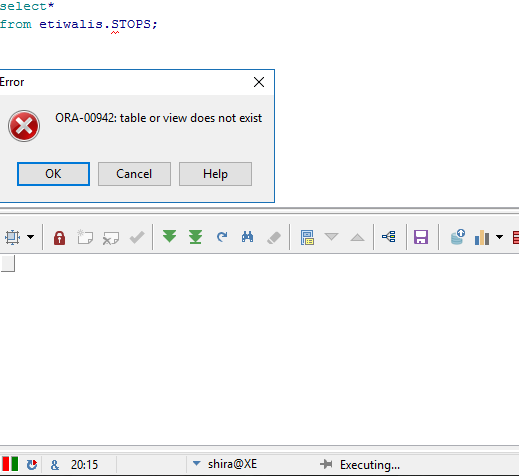
**GRANT&REVOKE:**grant:

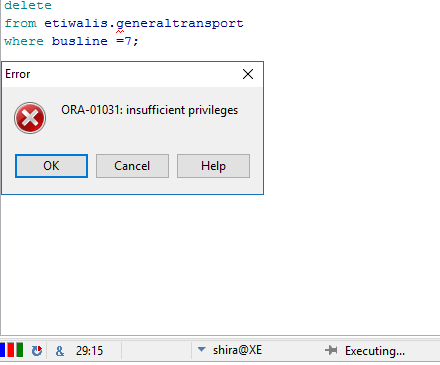
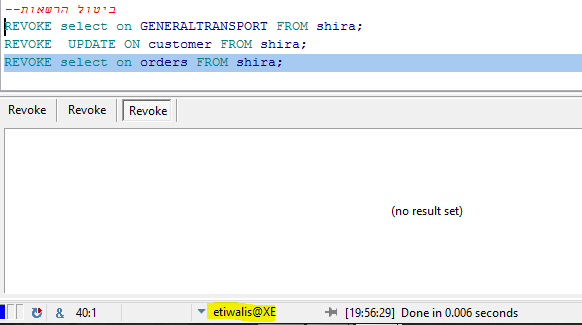


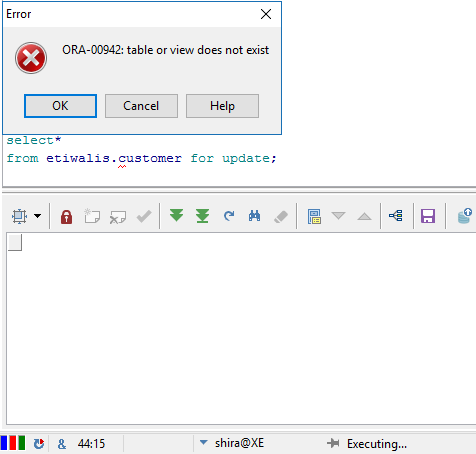
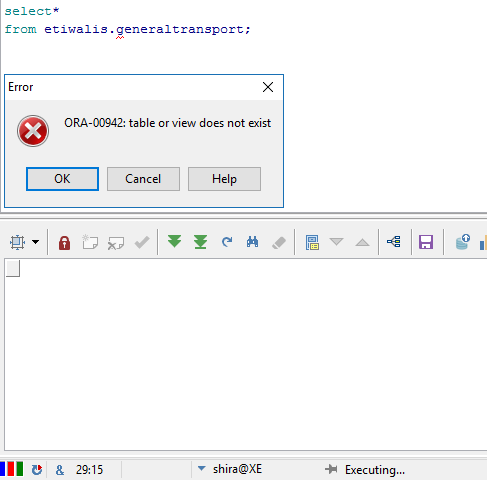
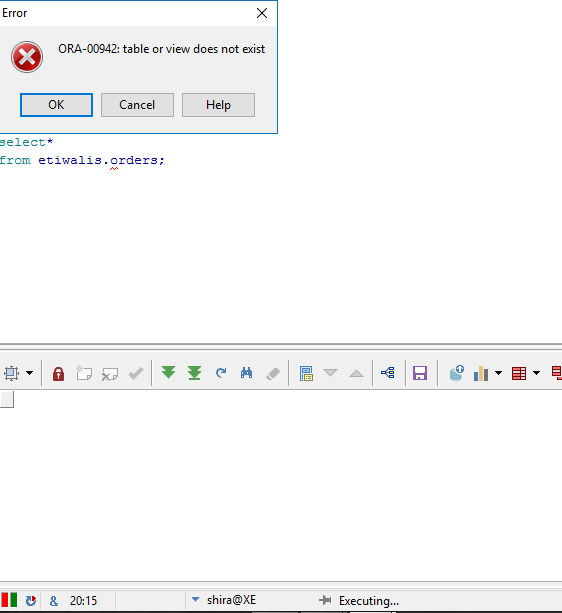
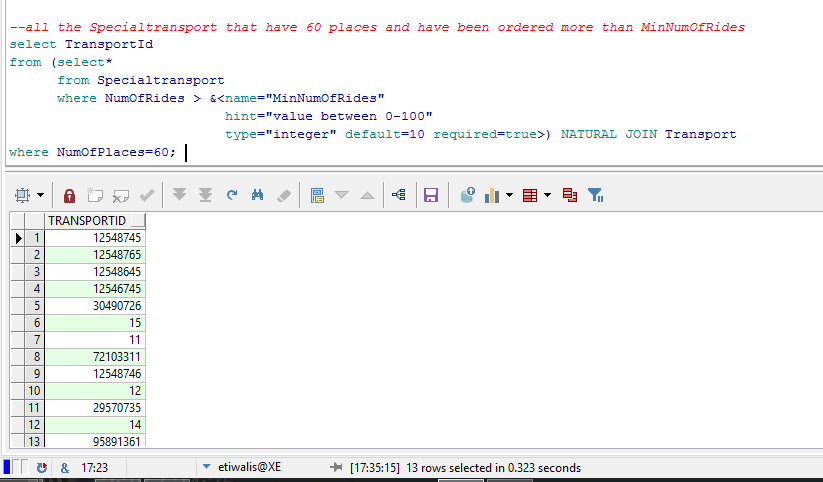
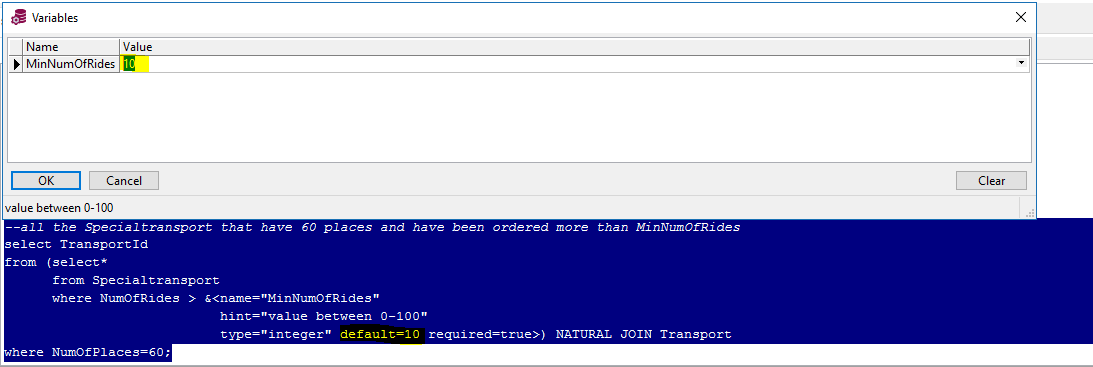
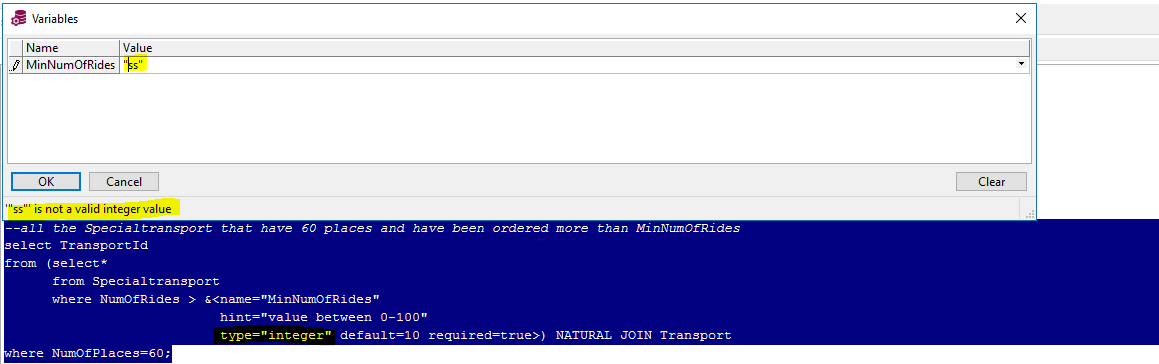
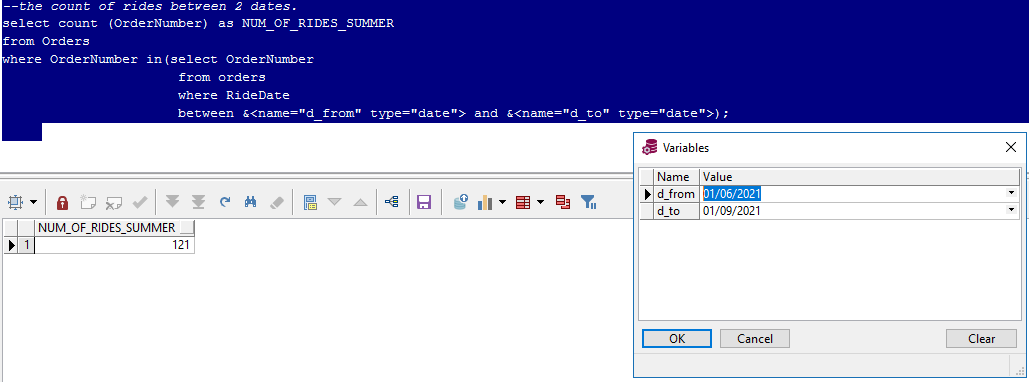
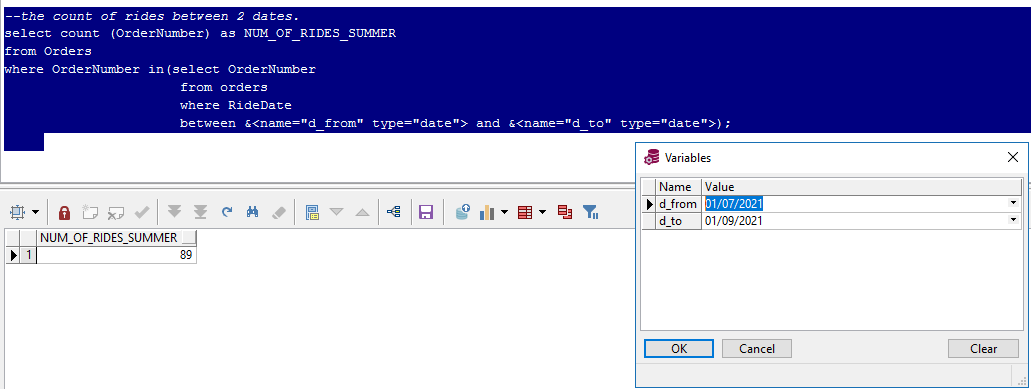


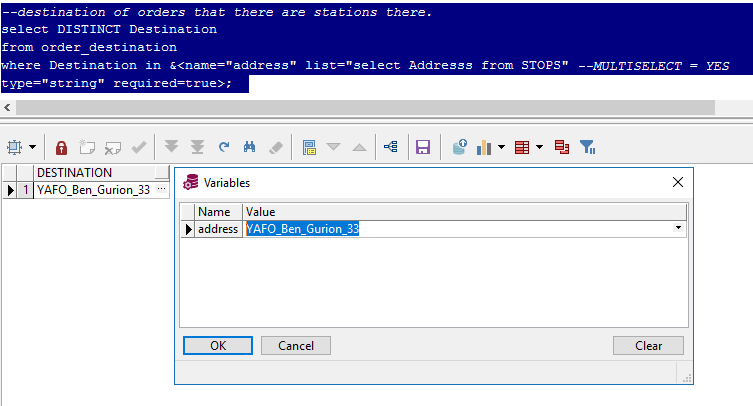
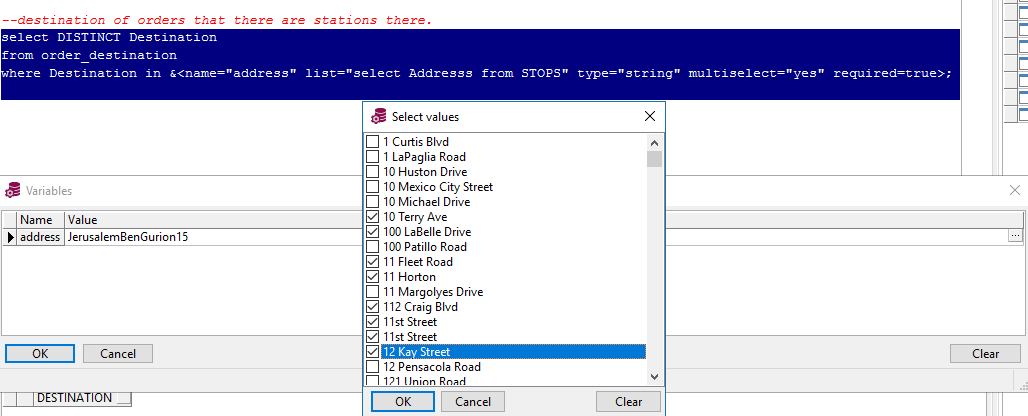
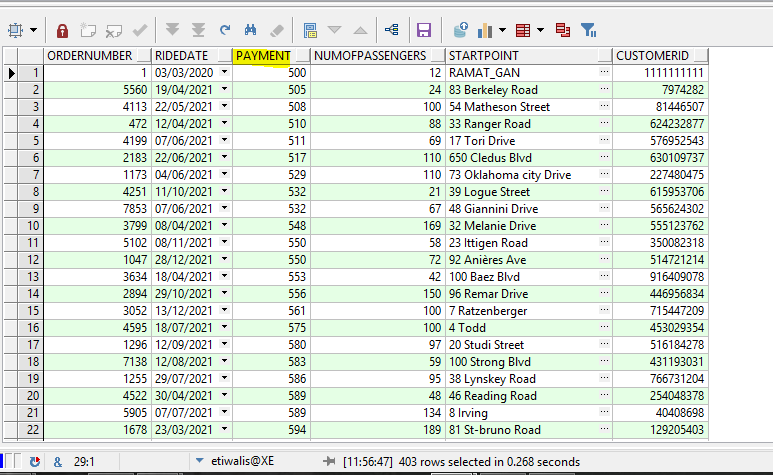
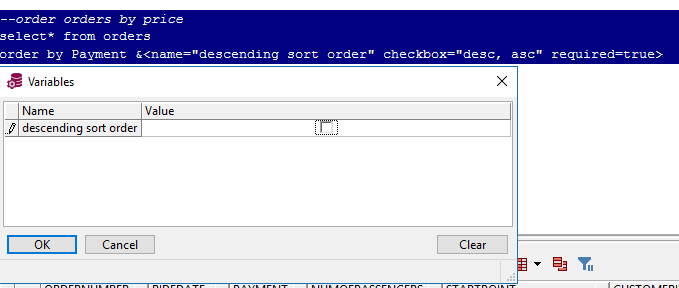
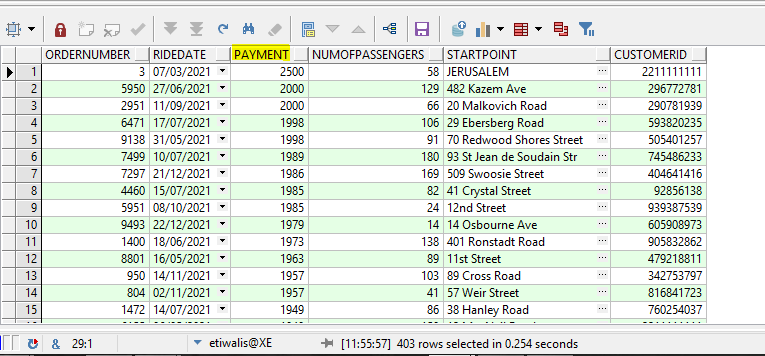
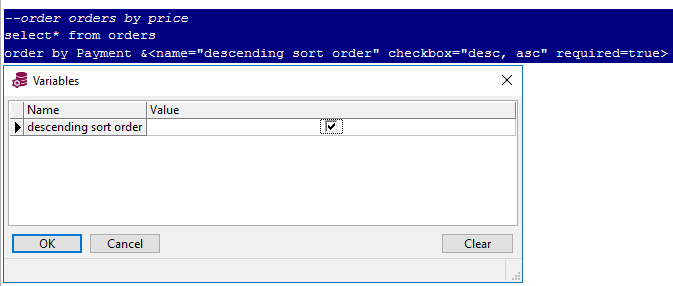
 

לא נותן כיון שלא ניתנה הרשאת גישה לטבלה זו.



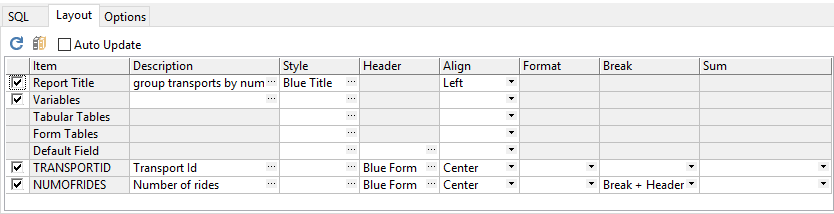
לא נותן כיון שלא ניתנה הרשאה למחיקה בטבלה זו.Revoke:

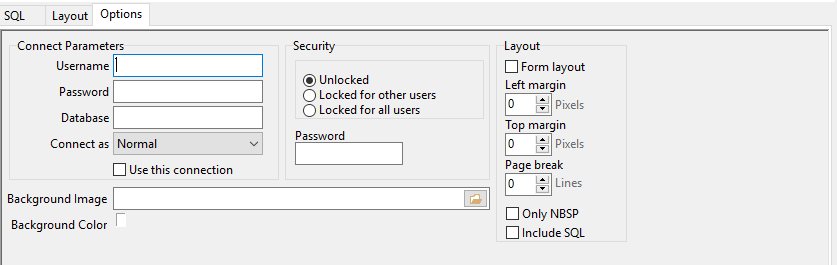
 **חלק ג'**כתיבת שאילתות עם פרמטרים: **1.**עבור ערך 12:  
ערך ברירת מחדל:  
עבור מחרוזת שאינה טיפוס מתאים: **2.** **3.**

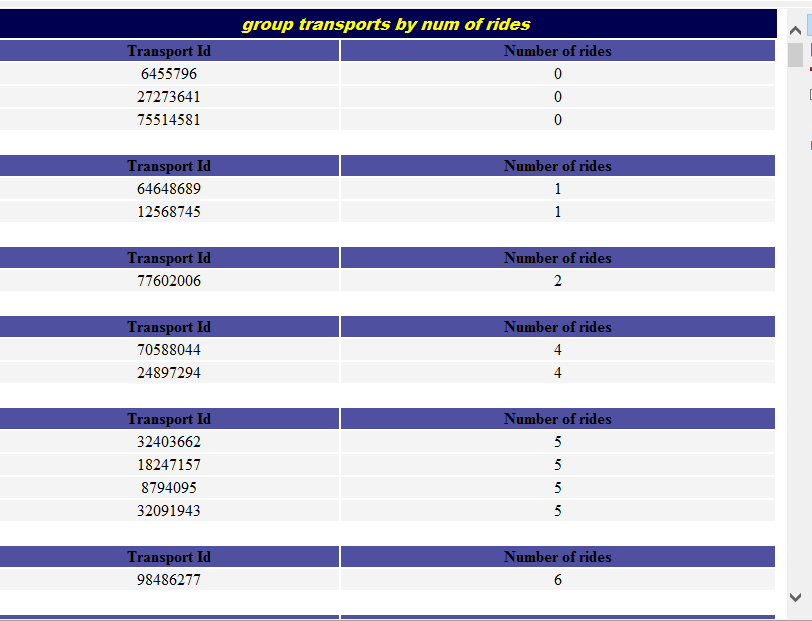
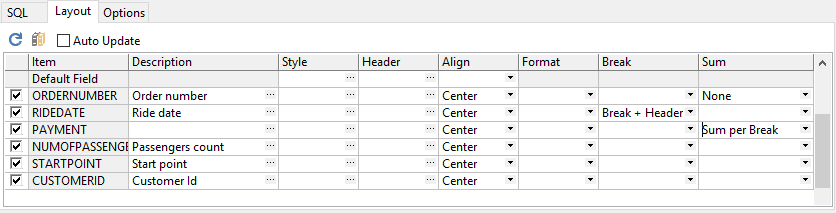
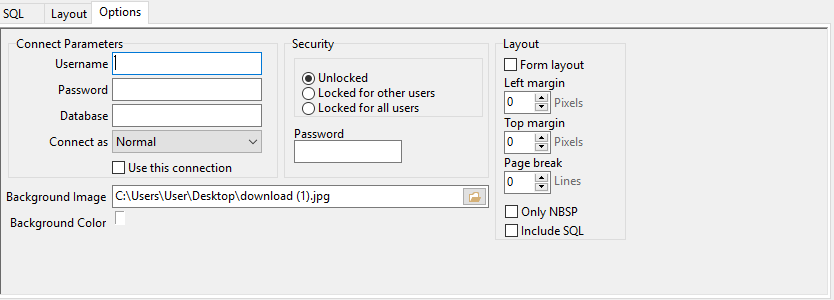
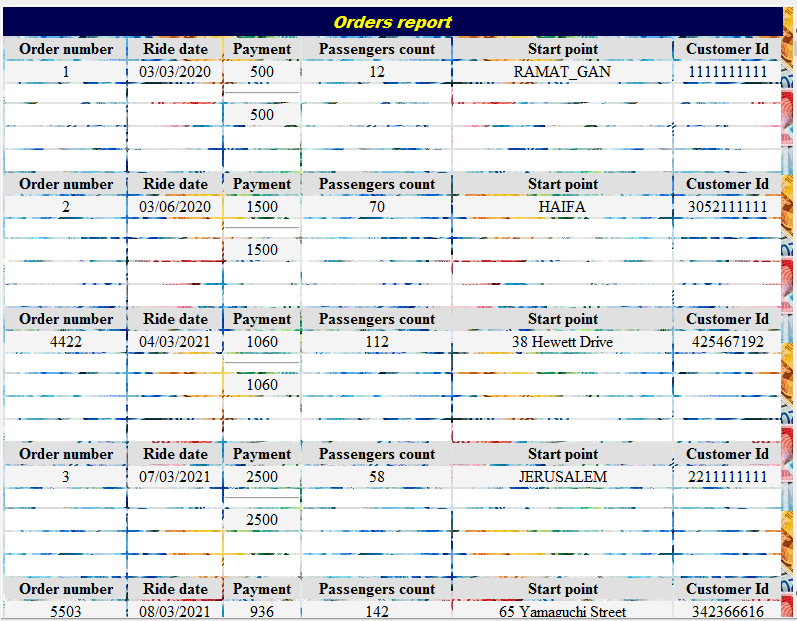
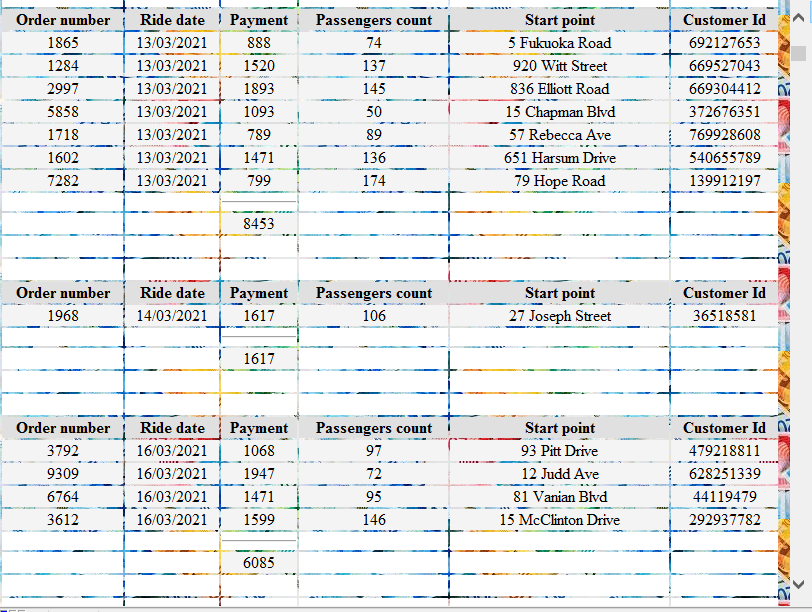
 **4.** **כתיבת דוחות:**

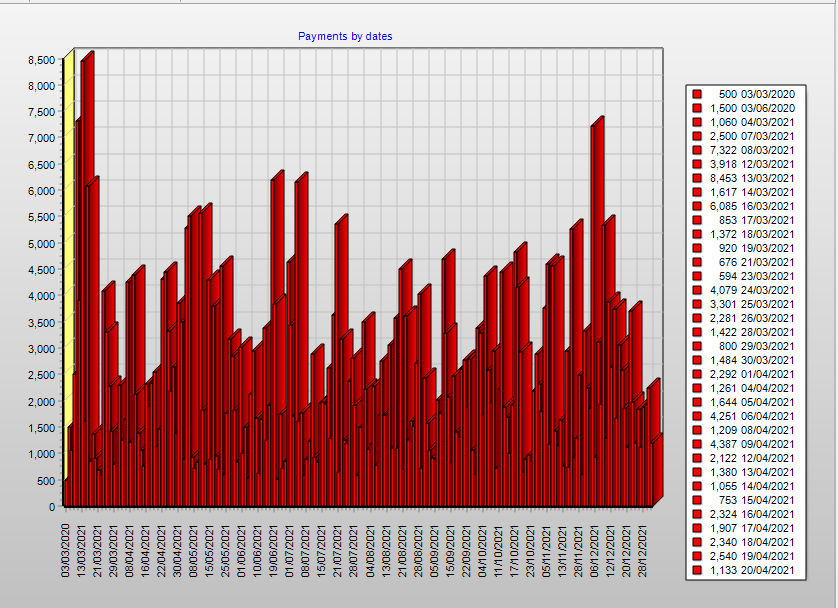
1. דוח המציג את כל האוטובוסים המיוחדים להזמנות לפי כמות הנסיעות של כל רכב.

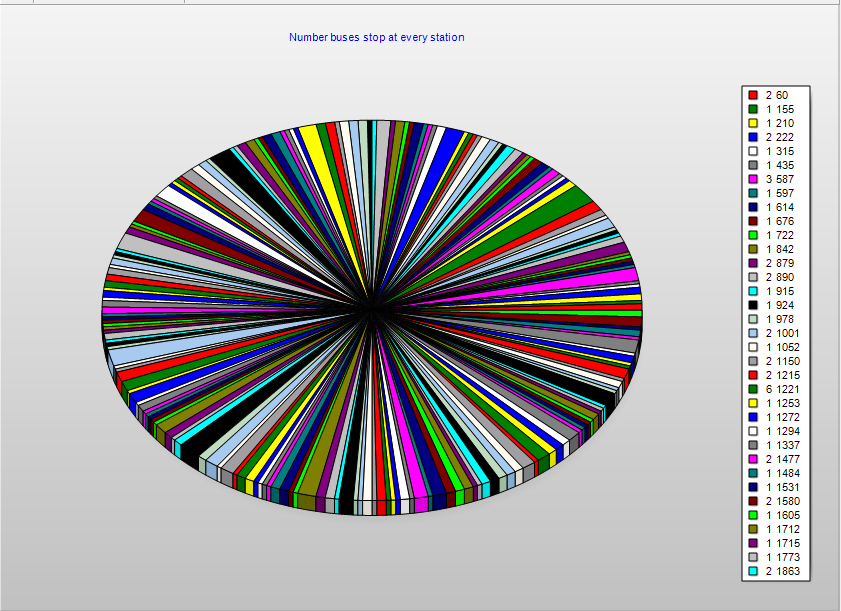


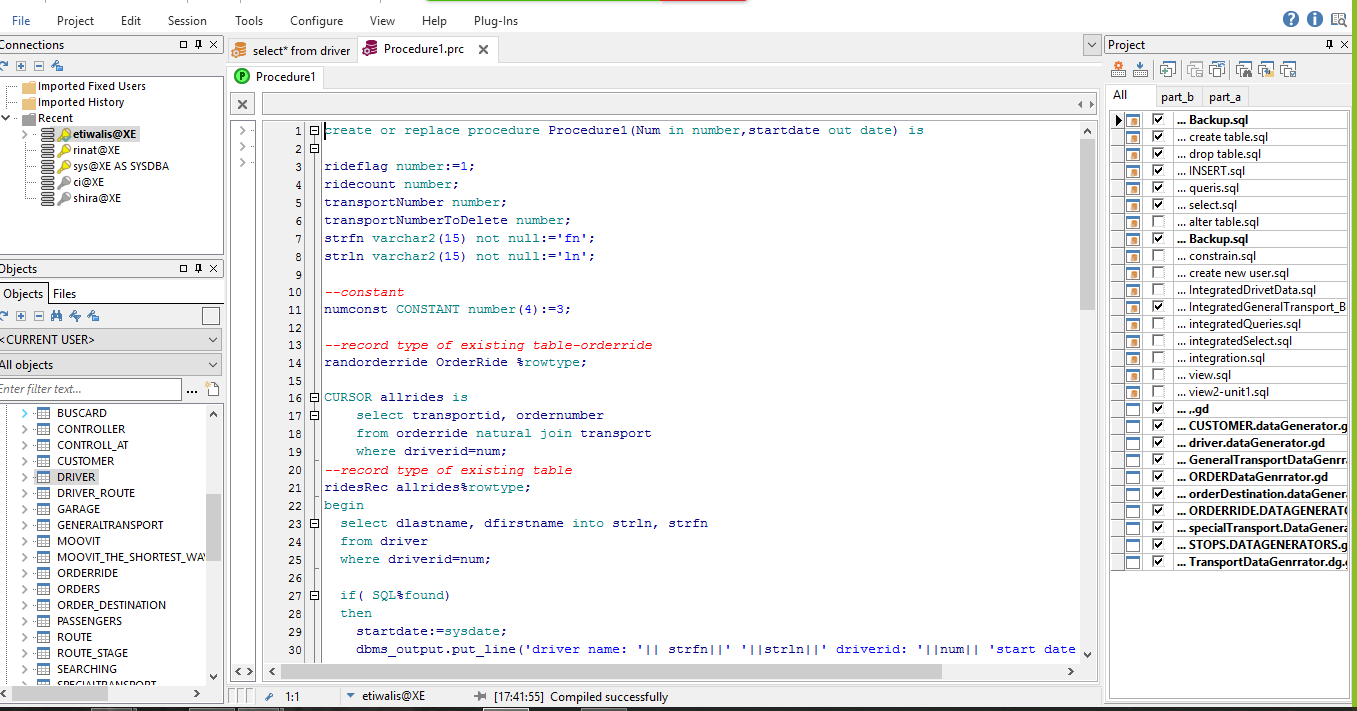


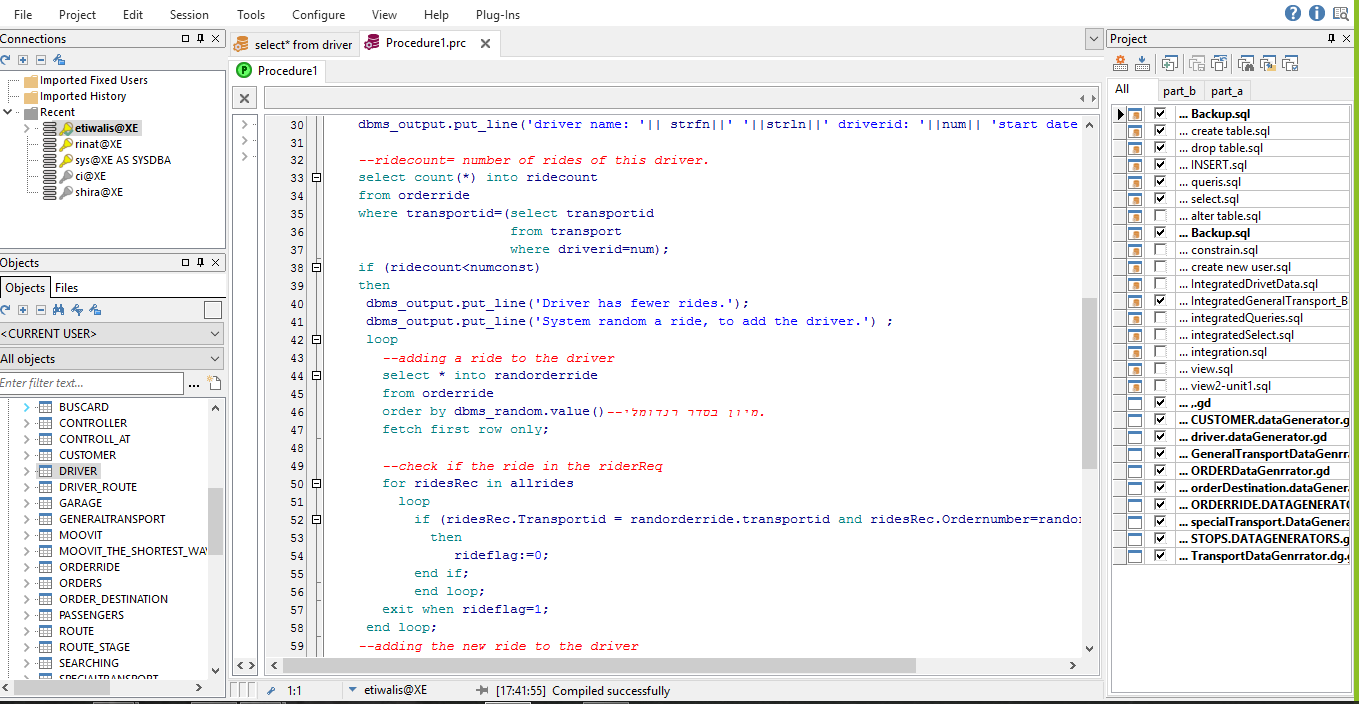


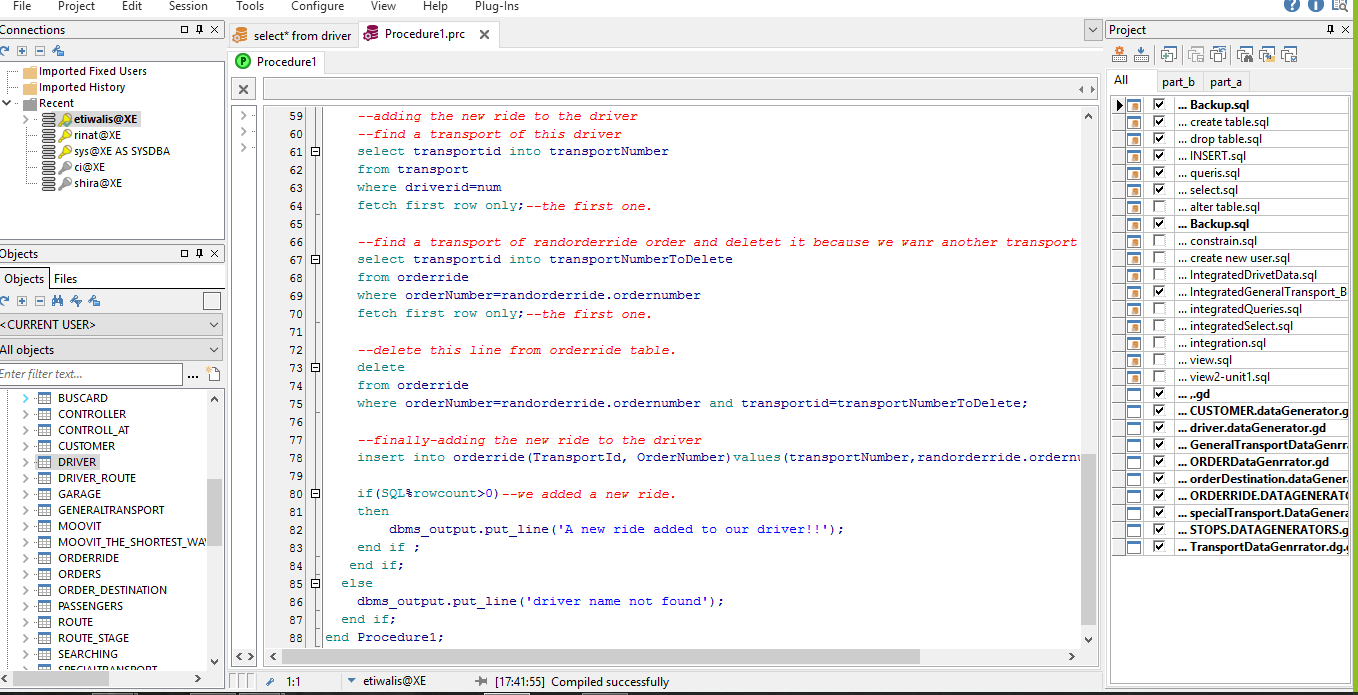
  
2. דו"ח המציג את כל ההזמנות מקובצות לפי תאריך ואת הסכום הכולל של הרווח מהן לכל תאריך.  
  
  
  
  


**גרפים:**1. גרף המציג את הרווחים לפי תאריכים (כמה הרווח הכולל מנסיעות מוזמנות בכל תאריך).

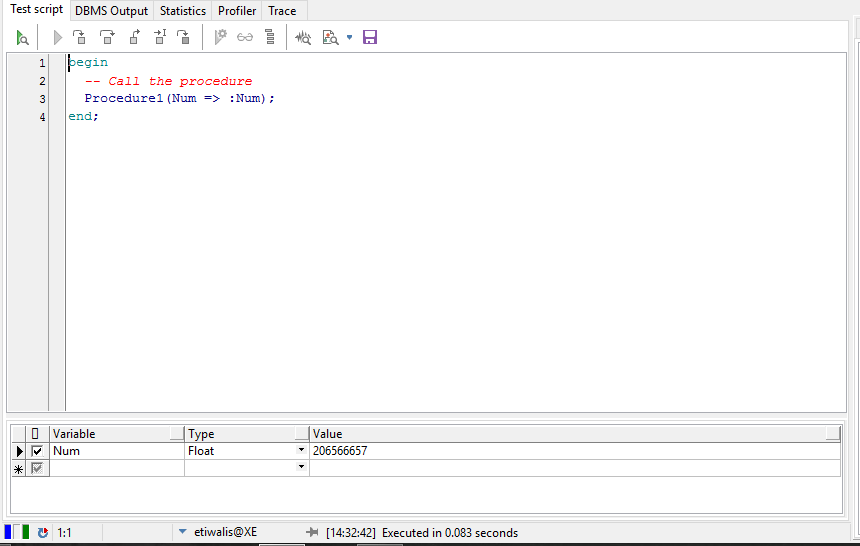
1. מספר האוטובוסים שעוצרים בכל תחנה.  
     
     
    **פונקציות ופרוצדורות:**

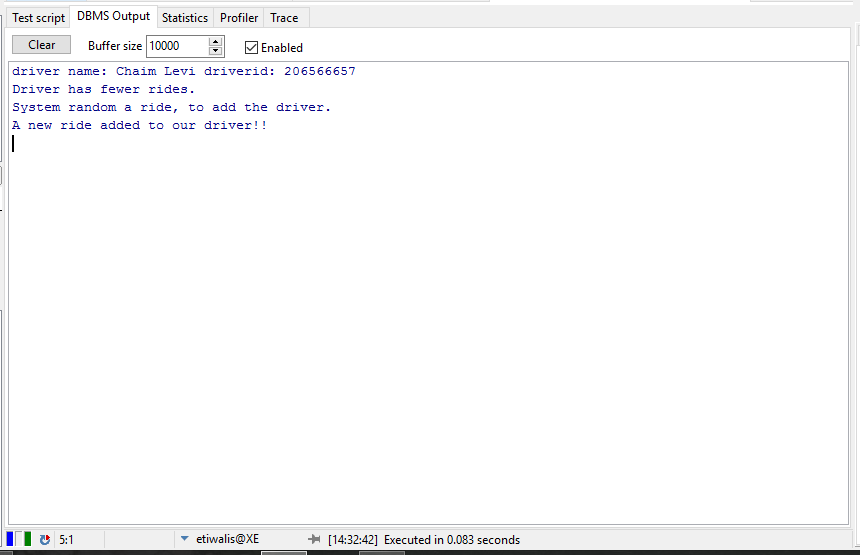
1.פרוצדורה שמקבלת מספר ת.ז של נהג ומחפשת אותו בבסיס הנתונים. אם קיים כזה נהג, היא בודקת אם יש לו פחות נסיעות ממספר קבוע מסוים, ואם כן מגרילה נסיעה רנדומלית ומעבירה אליו, במידה ולא שייכת אליו כבר.  




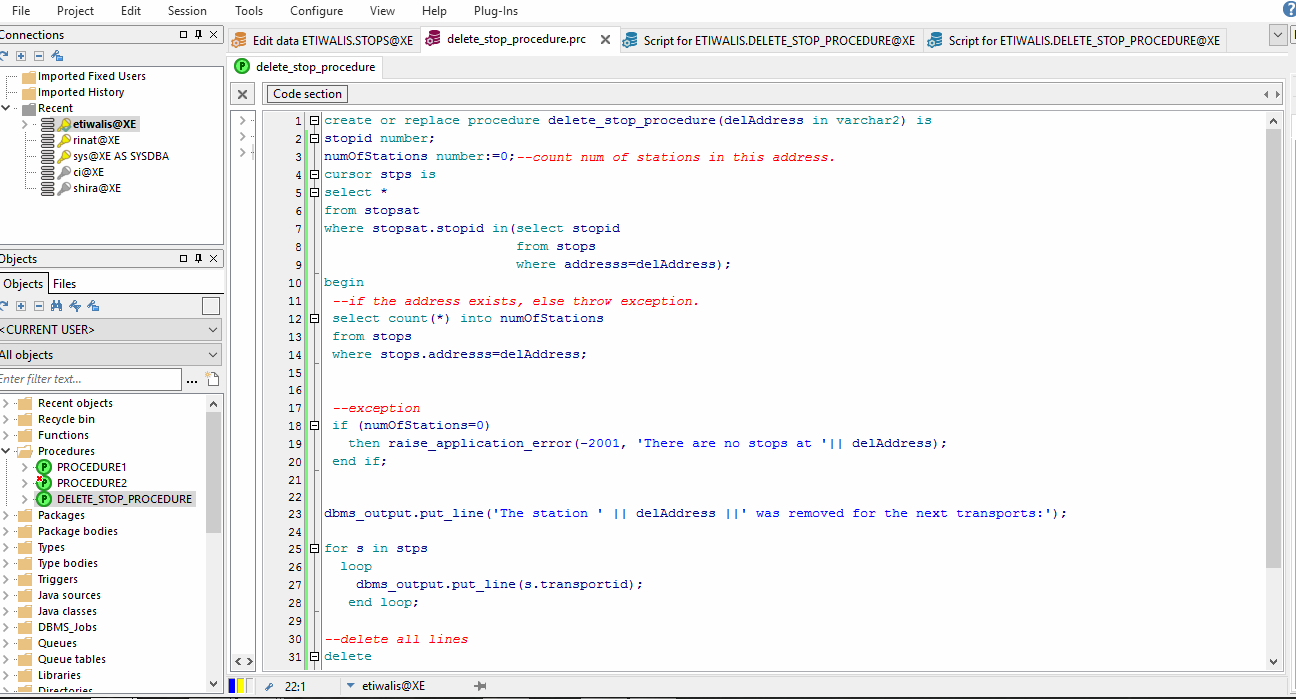


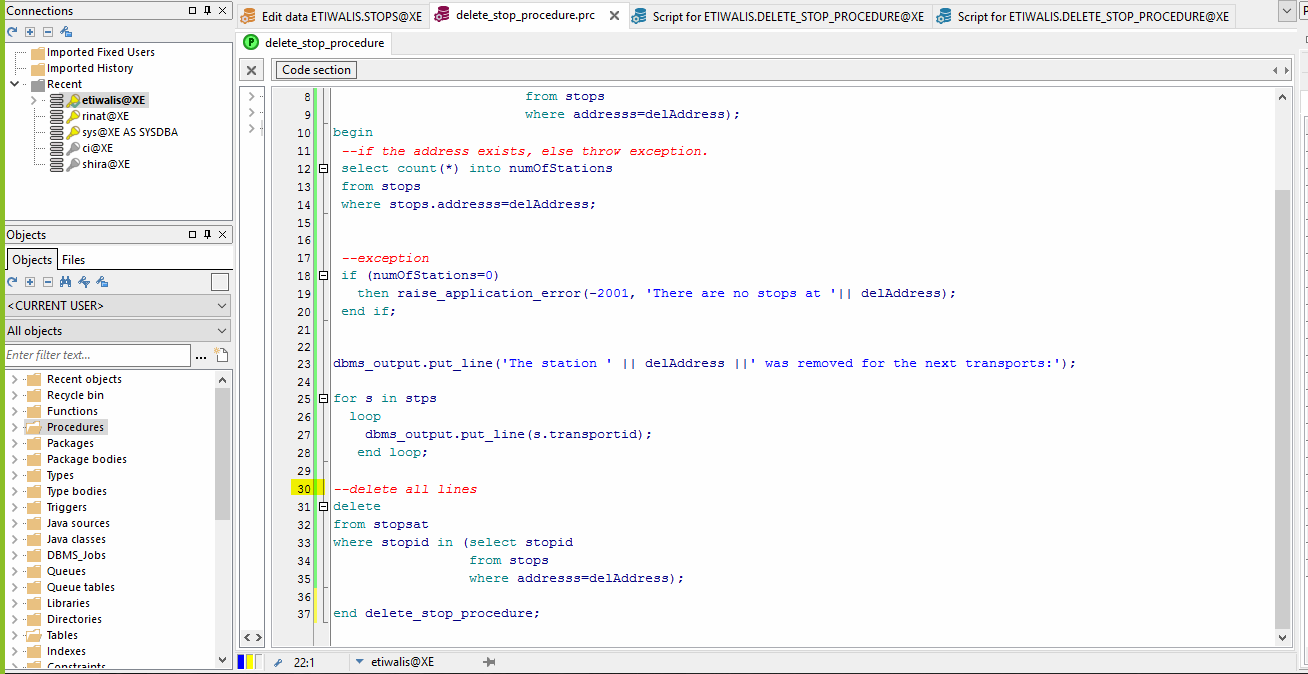
Test:

  
output:

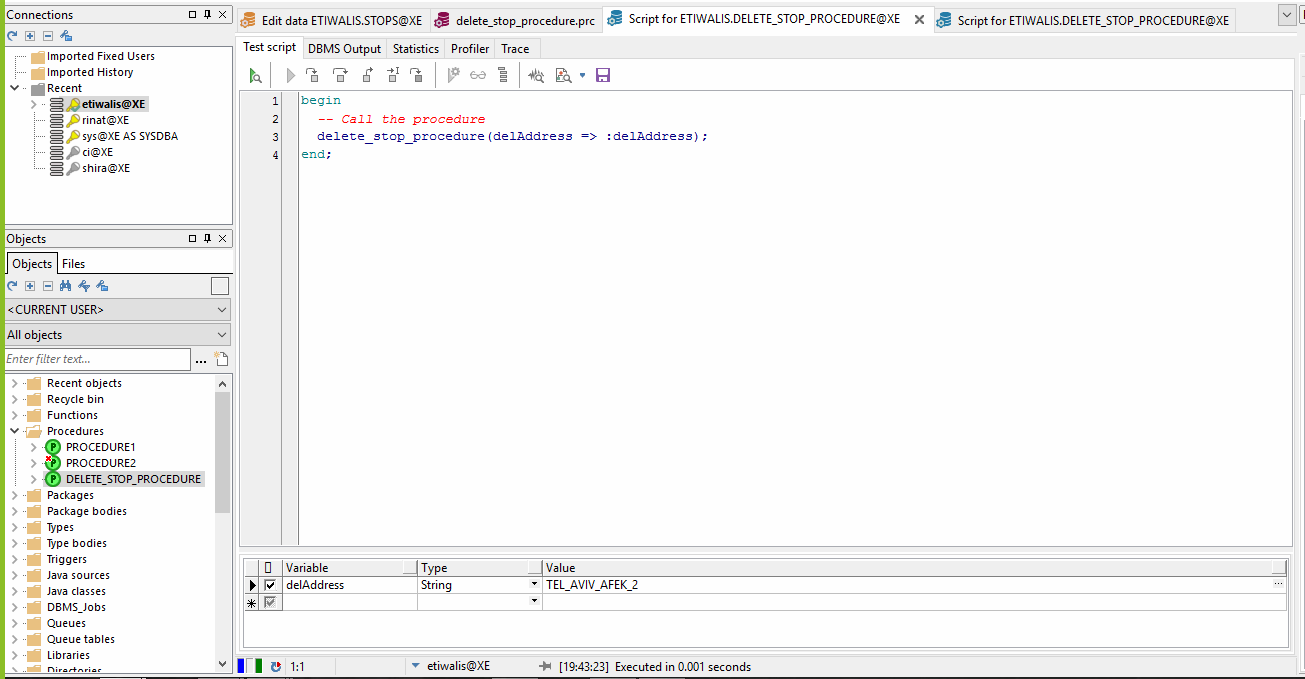


2. פרוצדורה עבור מחיקת תחנה. מקבלת כתובת ומוחקת את התחנה שבכתובת הזאת במידה וקיימת, אחרת זורקת חריגה.

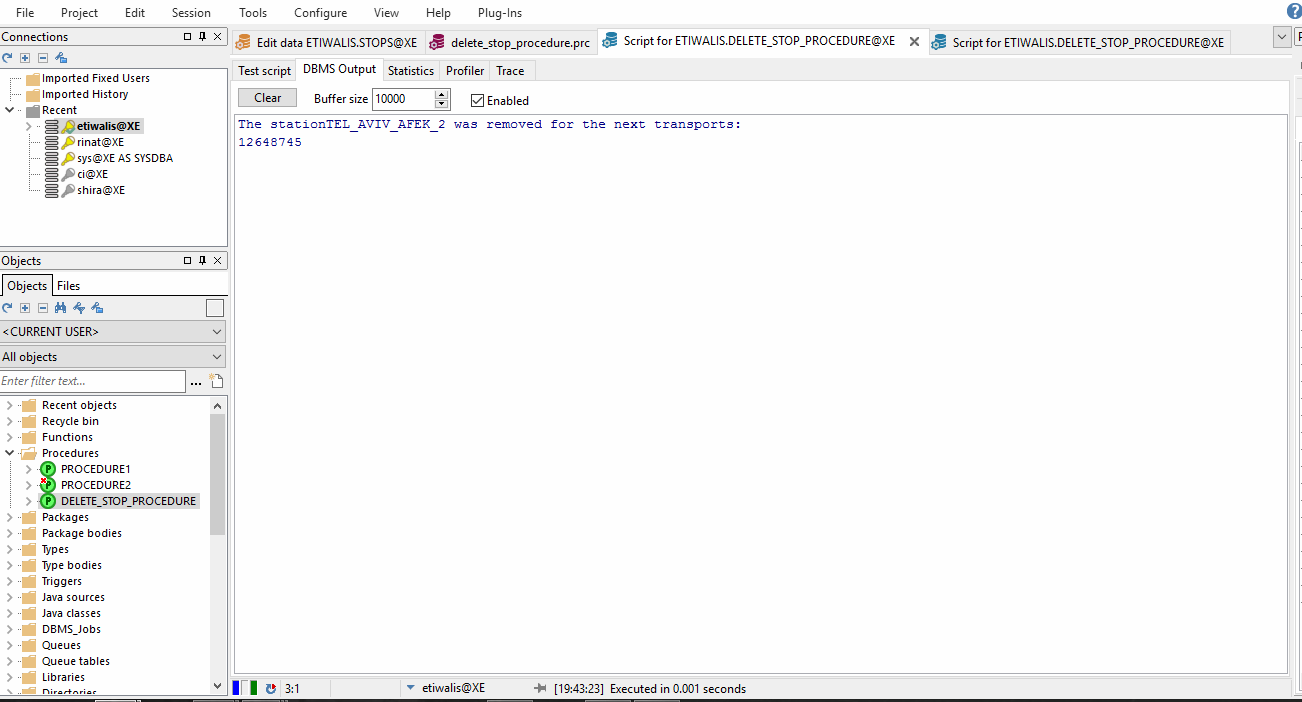




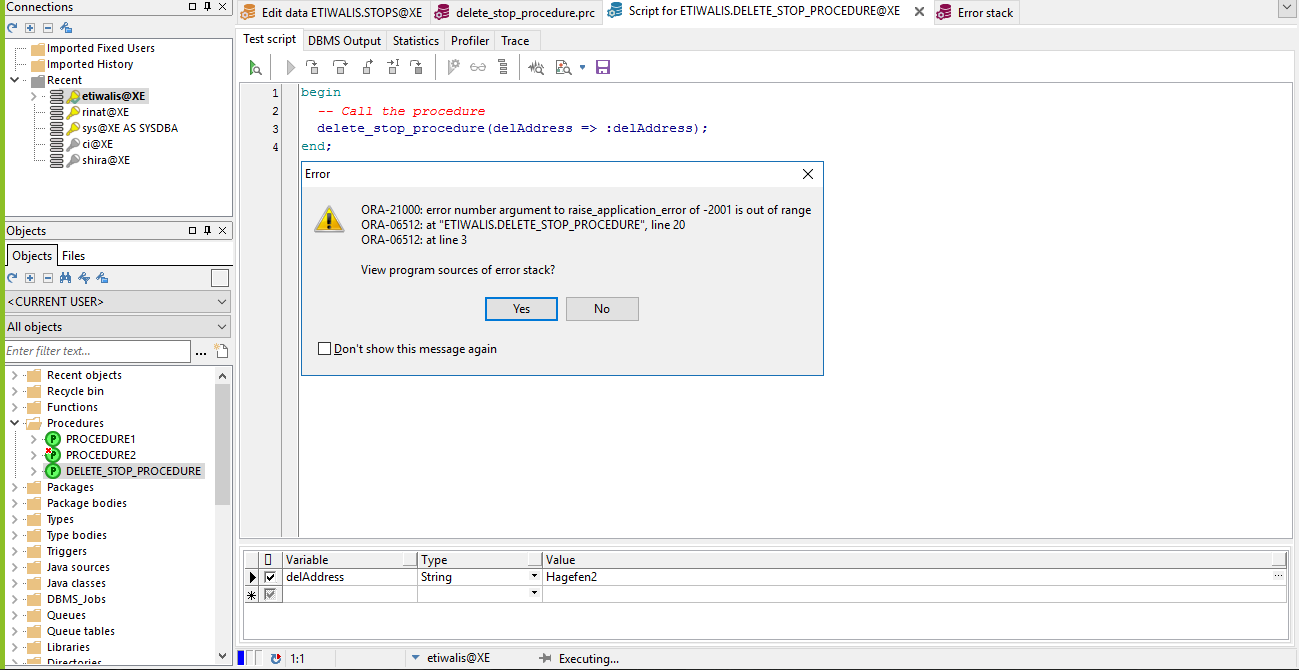
Test:



Output:

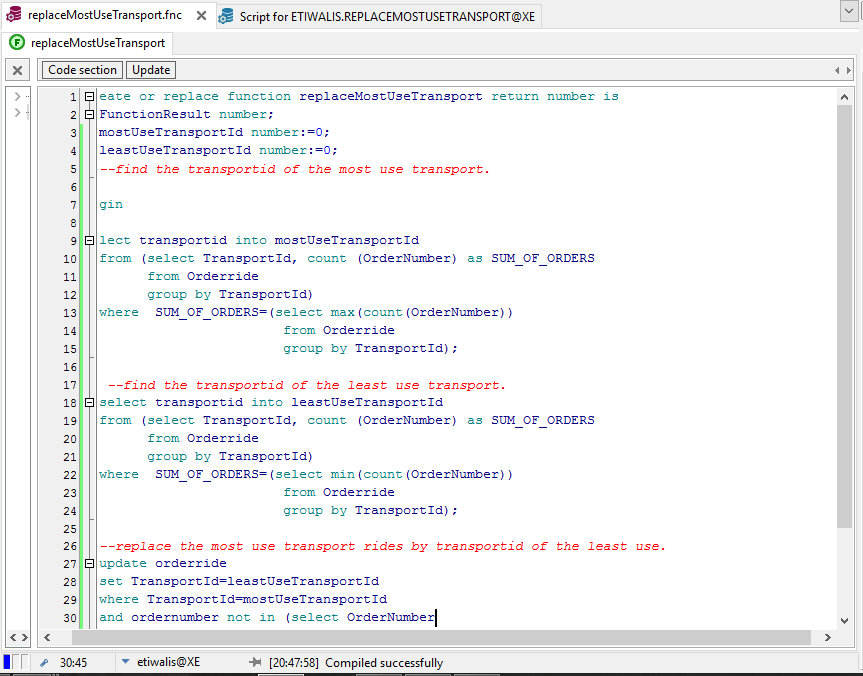


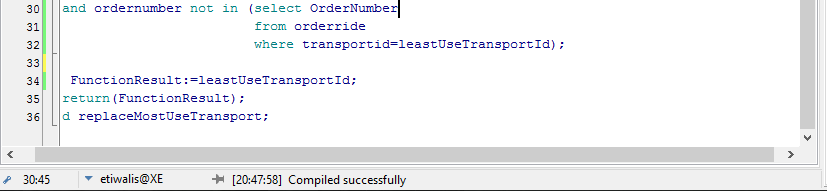
\*עבור כתובת שלא קיימת בה תחנה נזרקת חריגה.



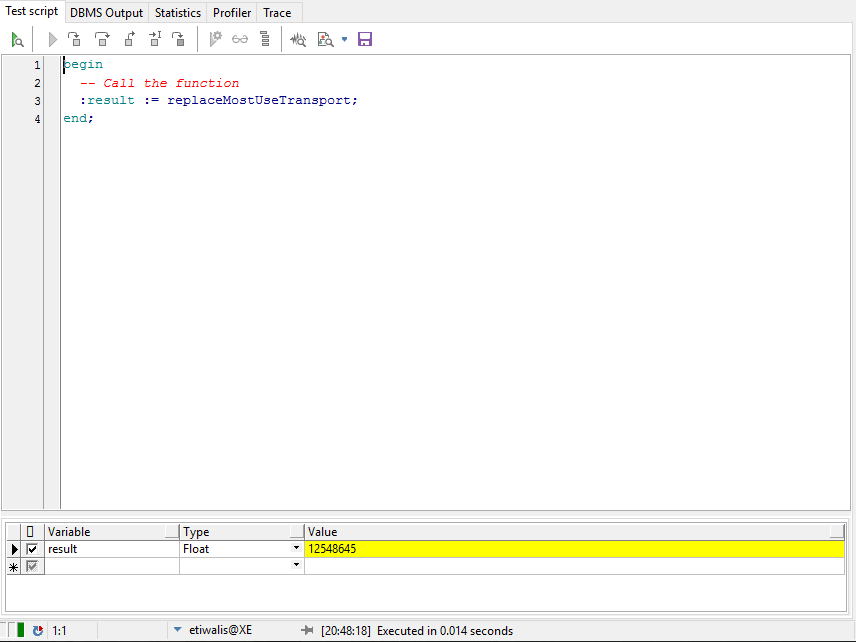
**פונקציות:**

1.פונקציה שמוצאת את האוטובוס שהוזמן הכי הרבה פעמים וכבר בלוי ומחליפה את הנסיעות שהוזמנו עבורו באוטובוס שהוזמן הכי פחות פעמים ולכן הכי פחות בלוי ומחזירה אותו.



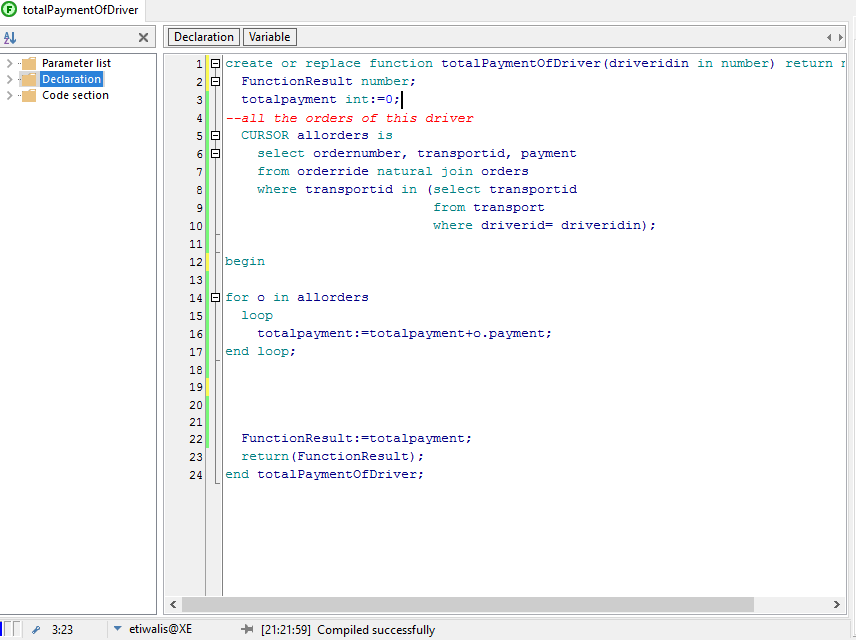


Output:

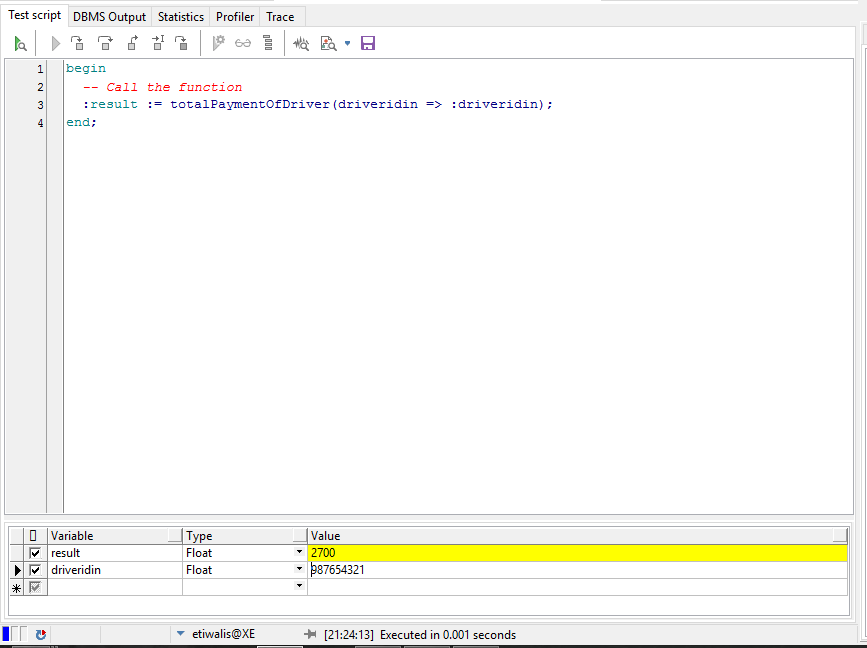


**2.**

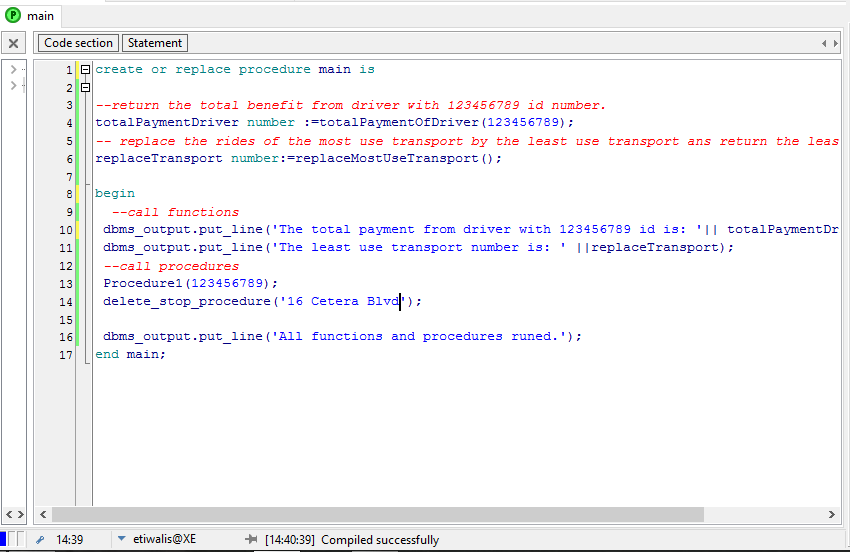
פונקציה שמקבלת ת.ז של נהג ומחזירה את הרווח הכולל של החברה מהנסיעות שהוא עשה.

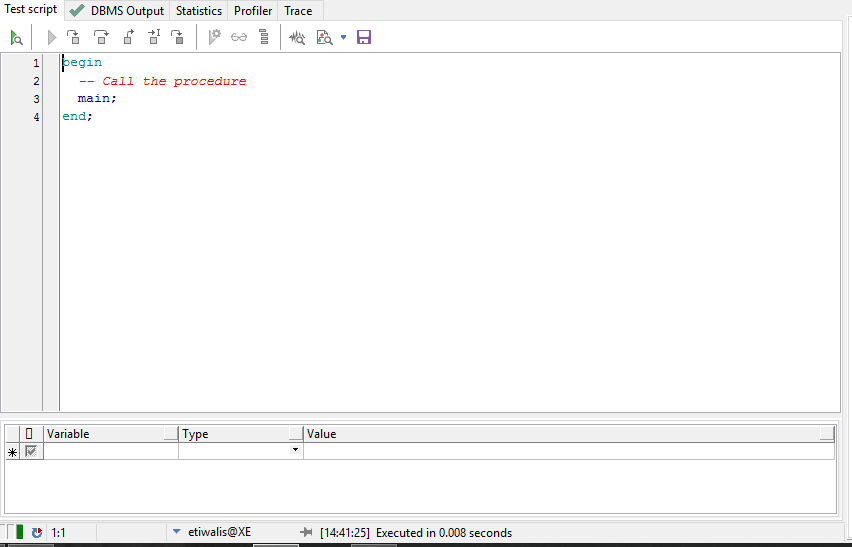


Test:

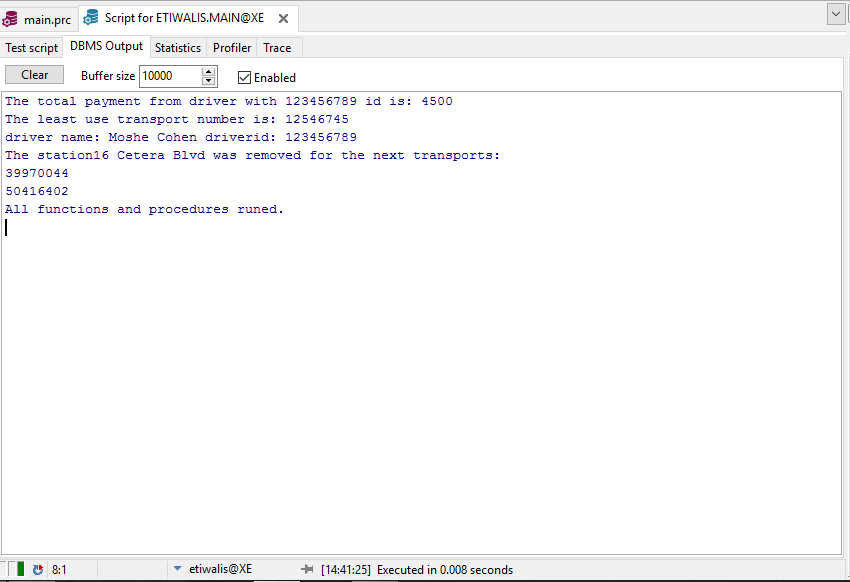


**Main:**

 **test:**



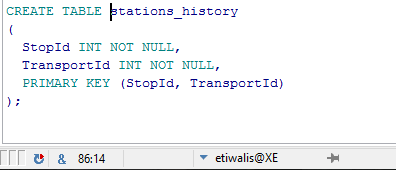
**Output:**

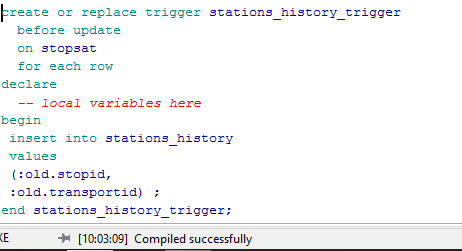


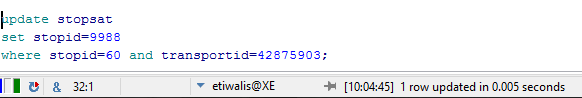
**טריגרים:**

1. לפני כל עדכון של תחנה (שינוי בתחנה או באוטובוס שעוצר בה) נשמור את השורה בטבלה שלפני העדכון, בטבלת ההיסטוריה של התחנות.

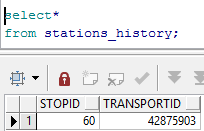
\*יצירת טבלת ההיסטוריה.

  
trigger:

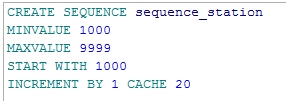
עדכון תחנה:



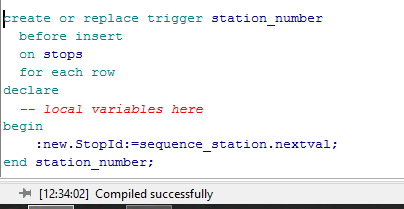
ניתן לראות שאכן התחנה נשמרה בטבלית ההיסטוריה כפי שהייתה לפני השינוי.



1. בכל הוספת תחנה חדשה, ייקבע לה מספר בצורה אוטומטית.  
   create sequence:



Trigger:



ניתן לראות שאכן כשהוספנו תחנות חדשות הן קיבלו מספרים מ1000והלאה לפי הסדר למרות שהכנסנו מספרים אחרים.  
