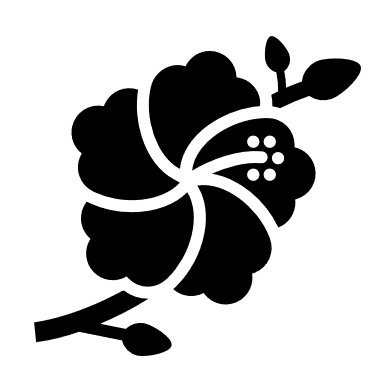
**דוח פרויקט**

**מיני פרויקט בבסיסי נתונים**

המערכת: מערכת לניהול ספריה

היחידה הנבחרת: ניהול אירועים בספריה



שירה קהלני 325283026

הלל אישון 212455570

תוכן עניינים

(שלב א')

* עמוד 3 - מבוא
* עמוד 4 - תרשים ERD
* עמוד 5 - תרשים DSD
* עמוד 6 - פקודות createTable
* עמוד 8 - פקודת desc
* עמוד 9 -יצירת נתונים ב data generator
* עמוד 10 -יצירת נתונים בקוד פייתון
* עמוד 12 -יצירת נתונים באתר Mockaroo
* עמוד 13 – גיבוי ושחזור

(שלב ב'- שאילתות)

* עמוד 14 – שאילתות select
* עמוד 18 – שאילתות delete
* עמוד 20 – שאילתות update
* עמוד 22 – שאילתות עם פרמטרים
* עמוד 26 - אילוצים

(שלב ג'- תכניות)

* עמוד 29 – תכנית א'
* עמוד 35 – תכנית ב'

(שלב ד'- אינטגרציה ומבטים)

* עמוד 36 – תהליך האינטגרציה
* עמוד 49 - מבטים

**- שלב א' -**

מבוא

המערכת הכללית היא מערכת לניהול ספריה. היחידה בה אנו מתמקדות היא ניהול האירועים בספריה.

מטעם הספרייה מתארגנים אירועים שונים ופעילויות שונות, בתאריכים שונים, באולמות שונים הקיימים בספריה, ולקהלי יעד שונים.

המערכת שלנו שומרת מידע רלוונטי אודות האירועים (בעבר או בעתיד), האולמות, מנהלי האירועים, המפעילים והמשתתפים.

מטרת המערכת היא לאפשר ניהול וסנכרון נוח ומסודר בין הגורמים השונים, וכן לאפשר מעקב והסקת מסקנות על סמך נתוני עבר, על מנת לייעל את מדור הפעילויות של הספרייה.

נרצה לתשאל את המערכת שאלות כגון:

כל המידע אודות פעילות מסוימת (פרטי המיקום, המנהל, המשתתפים, מפעיל וכו')

באיזה אולם שביעות הרצון מההפעלות היא הטובה ביותר – על פי ציון המשוב שסיפקו המשתתפים.

מהו האירוע שגרף את הרווחים המקסימליים (נחשב לפי כמות המשתתפים, המחיר לכרטיס, ומחיר המפעיל)

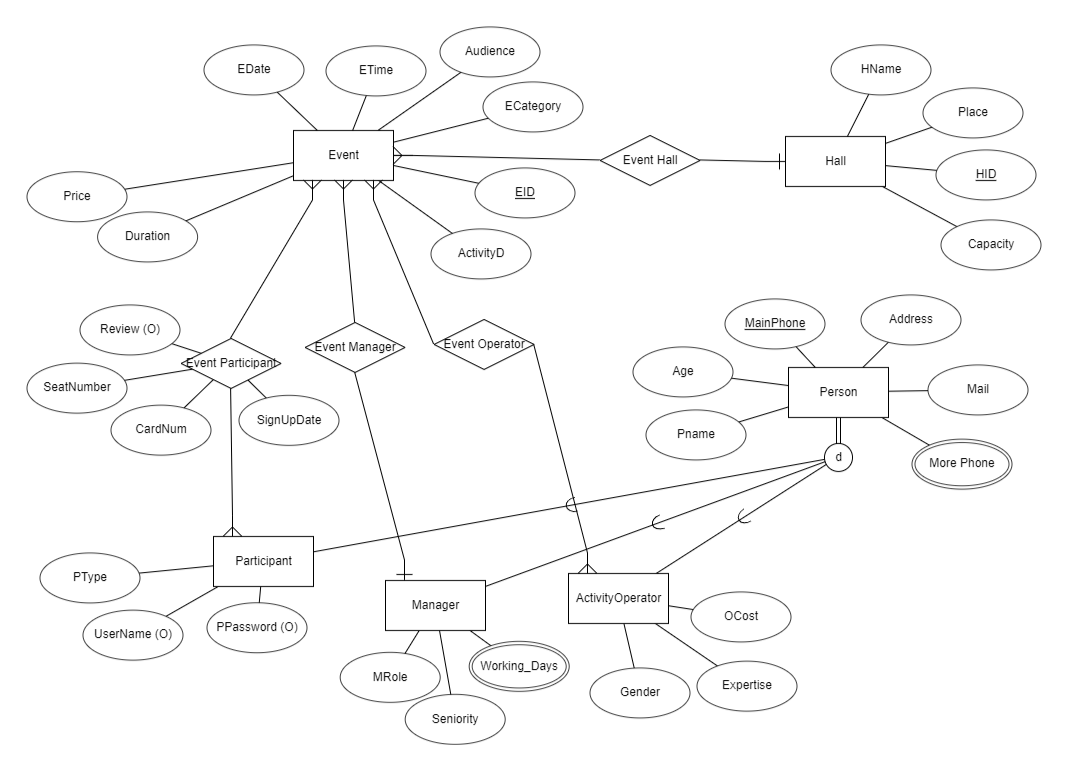
איזה סוג פעילות הוא המועדף בקרב ילדים מתחת לגיל 15 (נראה עבור איזה פעילות כמות הנרשמים מטווח גילאים זה היא הגבוהה ביותר)

מהו הגיל הממוצע של המשתתפים בהרצאות

חישוב הנחה עבור כל הנרשמים עד שבוע לפני יום האירוע

וכן הלאה...

תרשים ERD



נתייחס לכמה החלטות עיצוב שקיבלנו:

-קשר "רבים לרבים" בין ישות מפעיל לישות אירוע: מאחר שיתכן שיוזמן יותר ממפעיל יחיד לאירוע מסוים, הקשר הוא כזה שמאפשר זאת.

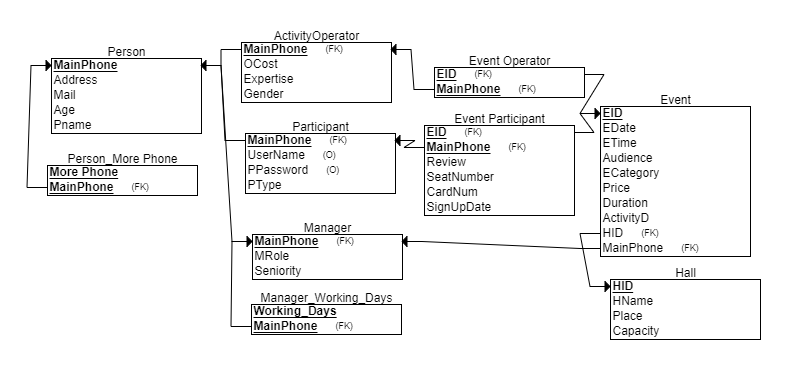
-לעומת זאת: קשר "יחיד לרבים" בין ישות נהל לישות אירוע: בחרנו לאפשר שיבוץ של מנהל אחד בלבד לכל אירוע שיהיה האחראי הראשי, כדי לשמור על הגיון ועל סדר.

-משתתף יכול להיות מסוג "מנוי" או "לא מנוי". עבור משתתף מנוי בלבד נשמרים שם המשתמש וסיסמא, ולכן תכונות אלו אופציונאליות בלבד.

-עבור ישות מנהל ישנה תכונה ששומרת את הימים בשבוע בהם הוא עובד, משום שדבר זה תורם להסקת מסקנות שקשורות לניהול אירועים, ולשיבוץ תקין של מנהל ואירוע זה לזה.

-מאחר ויש לנו שלוש ישויות (מפעיל, מנהל ומשתתף) שמייצגות אנשים, ארגנו את הקשרים בעזרת ירושה, כך שיש ישות "person" כללית, שלה תכונות הנוגעות לכל האנשים. וממנה יורשת כל ישות ספציפית יותר, ומכילה תכונות נוספות שרלוונטיות עבורה בלבד.

תרשים DSD



פקודת CREATE TABLE

CREATE TABLE Hall

(

HName varchar2(15) NOT NULL,

Place varchar2(40) NOT NULL,

HID INT NOT NULL,

Capacity INT NOT NULL,

PRIMARY KEY (HID)

);

CREATE TABLE Person

(

Address varchar2(40) NOT NULL,

Mail varchar2(30) NOT NULL,

MainPhone varchar2(10) NOT NULL,

Age INT NOT NULL,

Pname varchar2(30) NOT NULL,

PRIMARY KEY (MainPhone)

);

CREATE TABLE Manager

(

MRole varchar2(20) NOT NULL,

Seniority INT NOT NULL,

MainPhone varchar2(10) NOT NULL,

PRIMARY KEY (MainPhone),

FOREIGN KEY (MainPhone) REFERENCES Person(MainPhone)

);

CREATE TABLE ActivityOperator

(

OCost number(8,2) NOT NULL,

Expertise varchar2(40) NOT NULL,

Gender varchar2(15) NOT NULL,

MainPhone varchar2(10) NOT NULL,

PRIMARY KEY (MainPhone),

FOREIGN KEY (MainPhone) REFERENCES Person(MainPhone)

);

CREATE TABLE Participant

(

UserName varchar2(40),

PPassword INT,

PType varchar2(30) NOT NULL,

MainPhone varchar2(10) NOT NULL,

PRIMARY KEY (MainPhone),

FOREIGN KEY (MainPhone) REFERENCES Person(MainPhone)

);

CREATE TABLE Person\_More\_Phone

(

More\_Phone varchar2(10) NOT NULL,

MainPhone varchar2(10) NOT NULL,

PRIMARY KEY (More\_Phone, MainPhone),

FOREIGN KEY (MainPhone) REFERENCES Person(MainPhone)

);

CREATE TABLE Manager\_Working\_Days

(

Working\_Days varchar2(10) NOT NULL,

MainPhone varchar2(10) NOT NULL,

PRIMARY KEY (Working\_Days, MainPhone),

FOREIGN KEY (MainPhone) REFERENCES Manager(MainPhone)

);

CREATE TABLE Event

(

EDate date NOT NULL,

ETime number(4,0) NOT NULL,

Audience varchar2(30) NOT NULL,

ECategory varchar2(40) NOT NULL,

ActivityD varchar2(300) NOT NULL,

Price NUMBER(6,2) NOT NULL,

Duration INT NOT NULL,

EID INT NOT NULL,

HID INT NOT NULL,

MainPhone varchar2(10) NOT NULL,

PRIMARY KEY (EID),

FOREIGN KEY (HID) REFERENCES Hall(HID),

FOREIGN KEY (MainPhone) REFERENCES Manager(MainPhone)

);

CREATE TABLE Event\_Operator

(

EID INT NOT NULL,

MainPhone varchar2(10) NOT NULL,

PRIMARY KEY (EID, MainPhone),

FOREIGN KEY (EID) REFERENCES Event(EID),

FOREIGN KEY (MainPhone) REFERENCES ActivityOperator(MainPhone)

);

CREATE TABLE Event\_Participant

(

Review INT NOT NULL,

SeatNumber INT NOT NULL,

CardNum INT NOT NULL,

SignUpDate date NOT NULL,

EID INT NOT NULL,

MainPhone varchar2(10) NOT NULL,

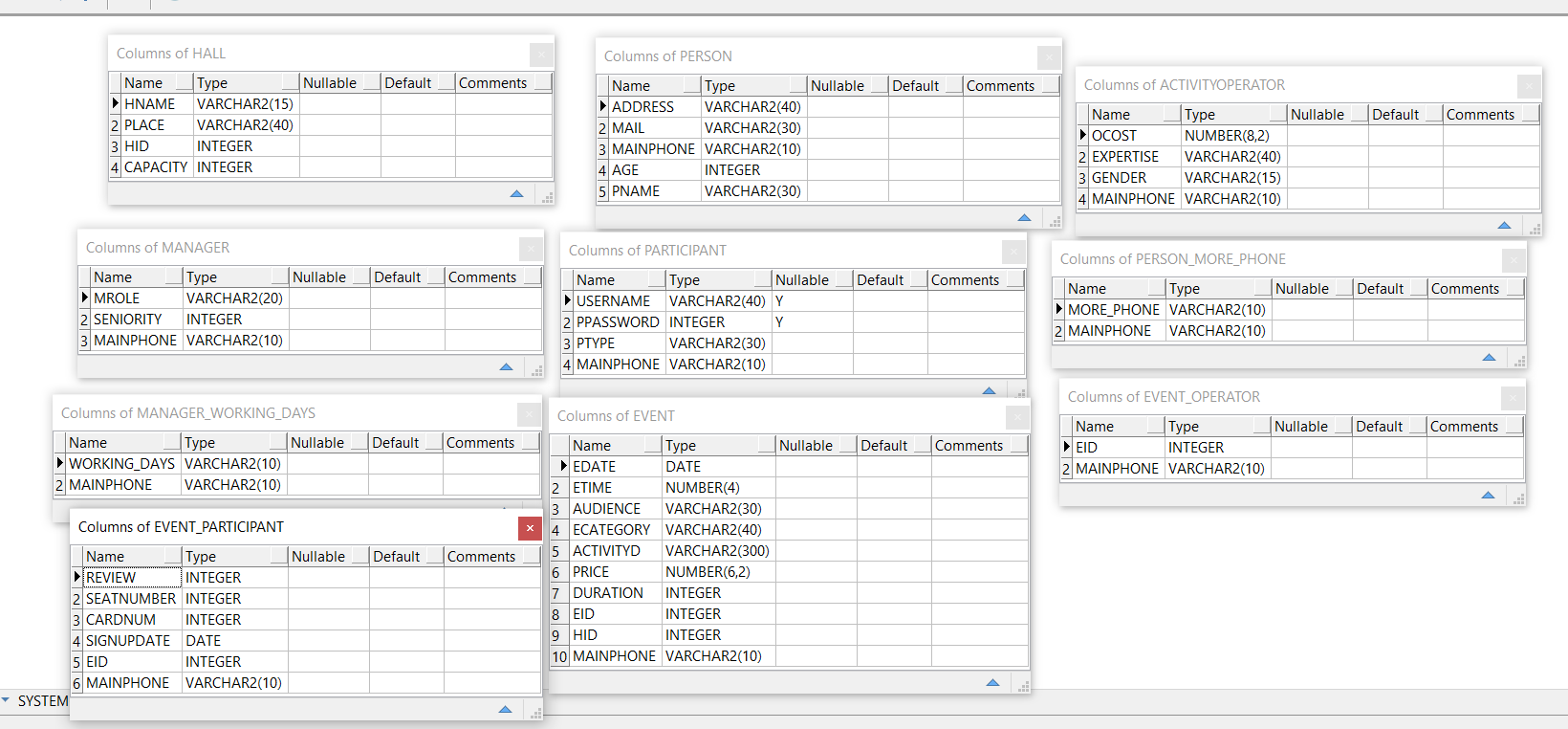
PRIMARY KEY (EID, MainPhone),

FOREIGN KEY (EID) REFERENCES Event(EID),

FOREIGN KEY (MainPhone) REFERENCES Participant(MainPhone)

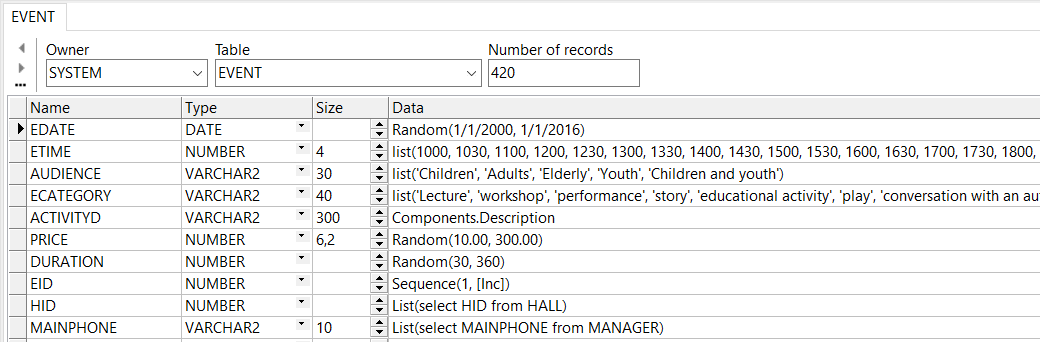
);

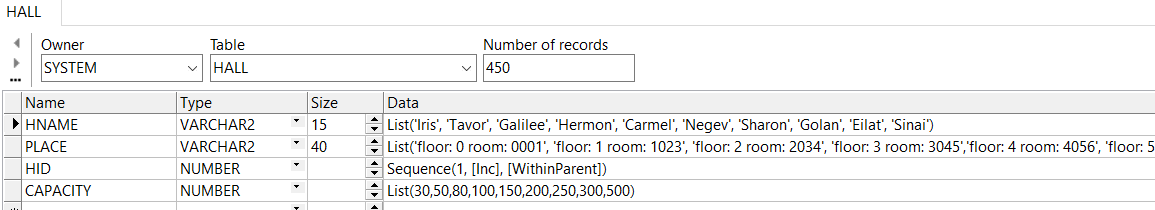
Desc

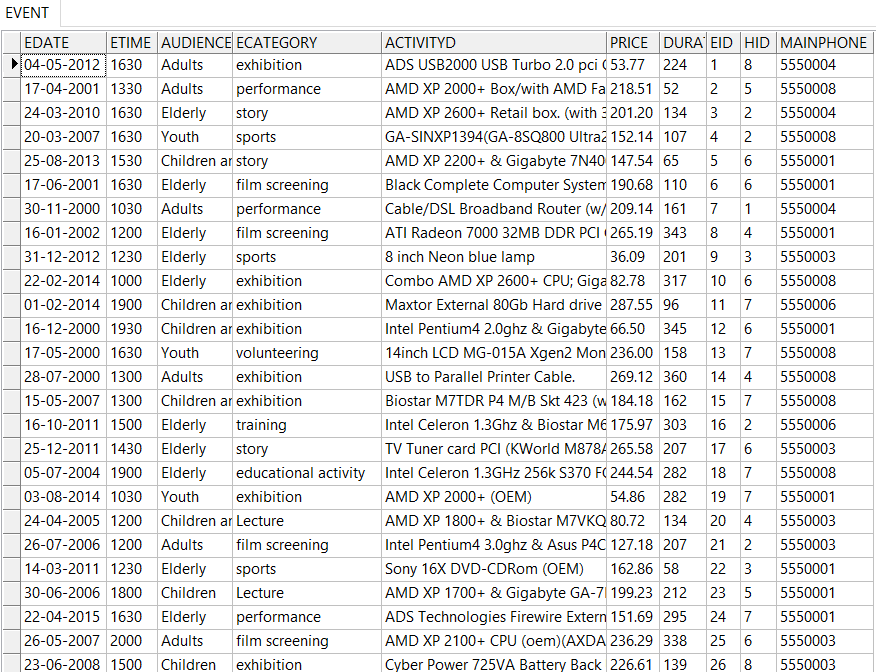
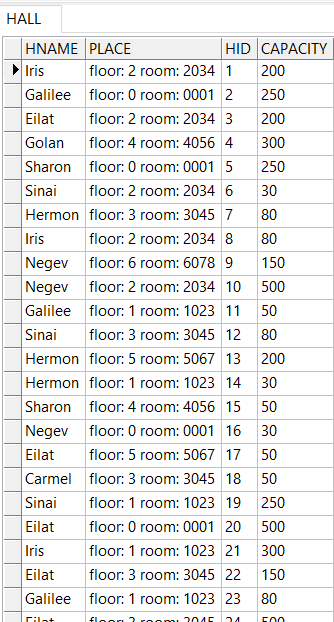


הכנסת נתונים ע"י data generator

לדוגמא:

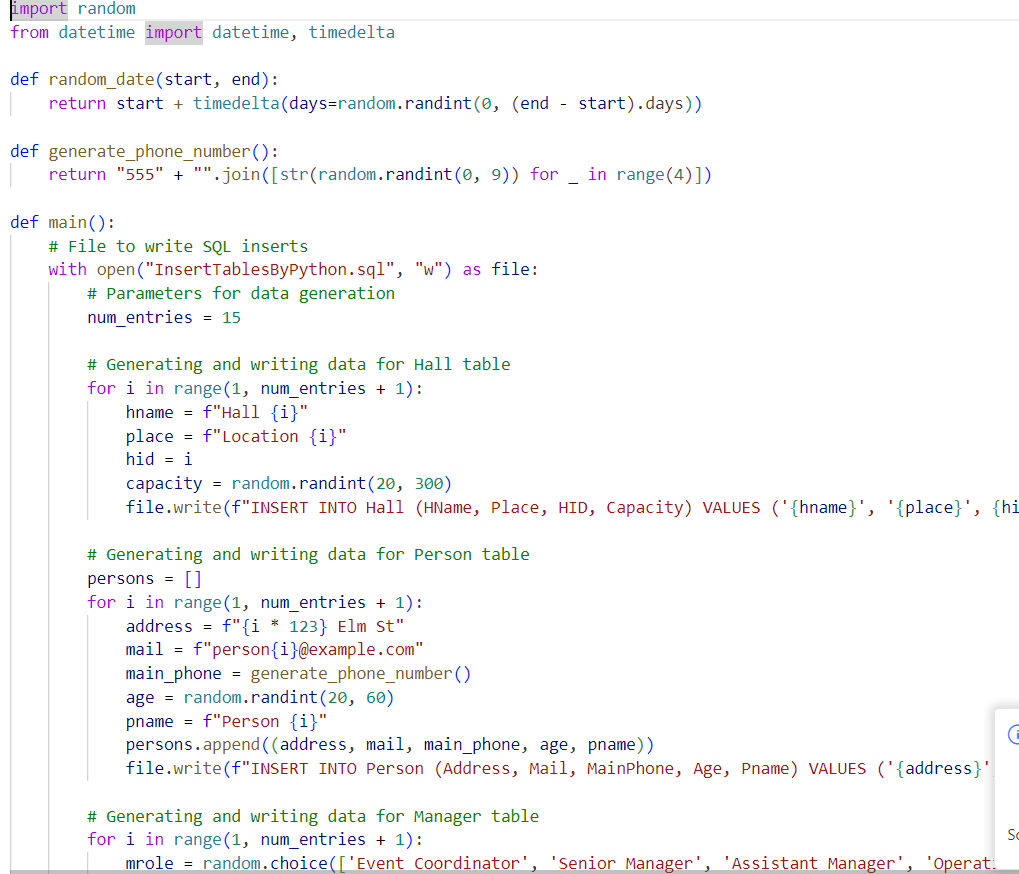






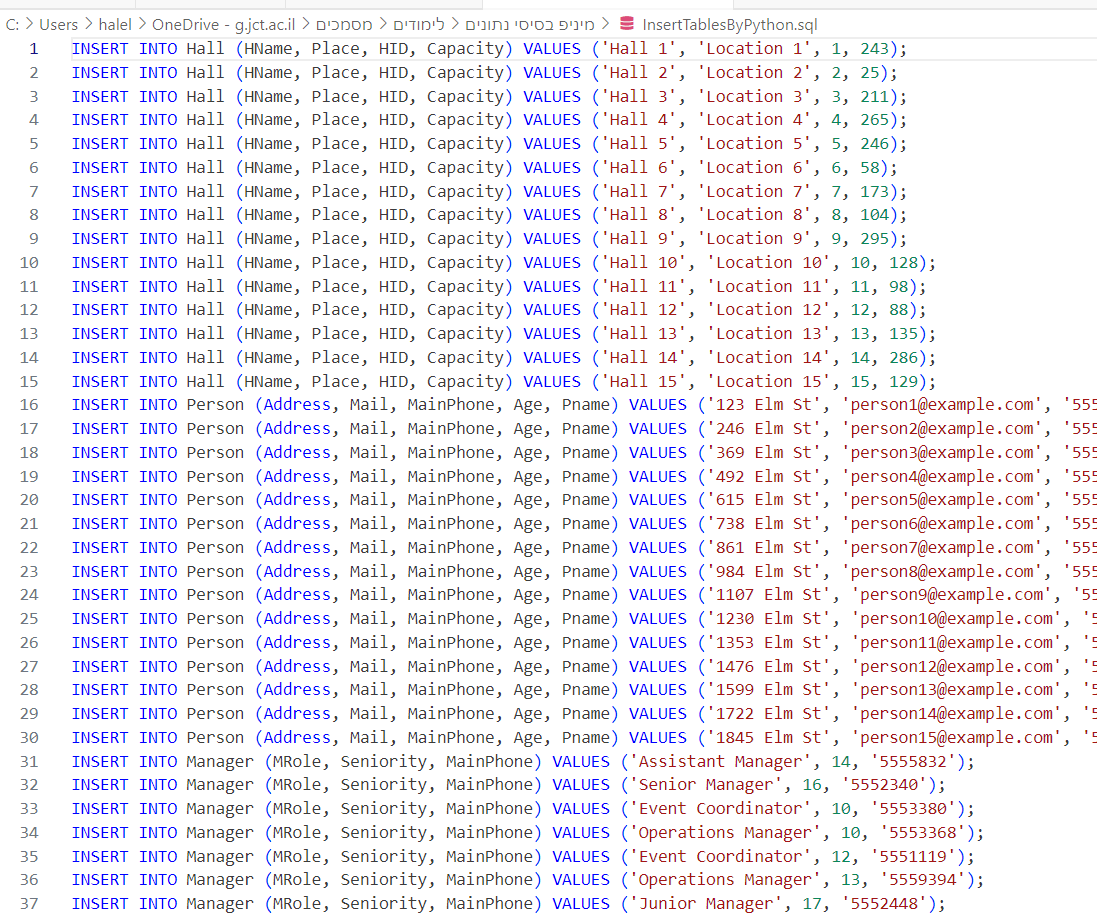
הכנסת נתונים עם קוד python

חלק מן הקוד:



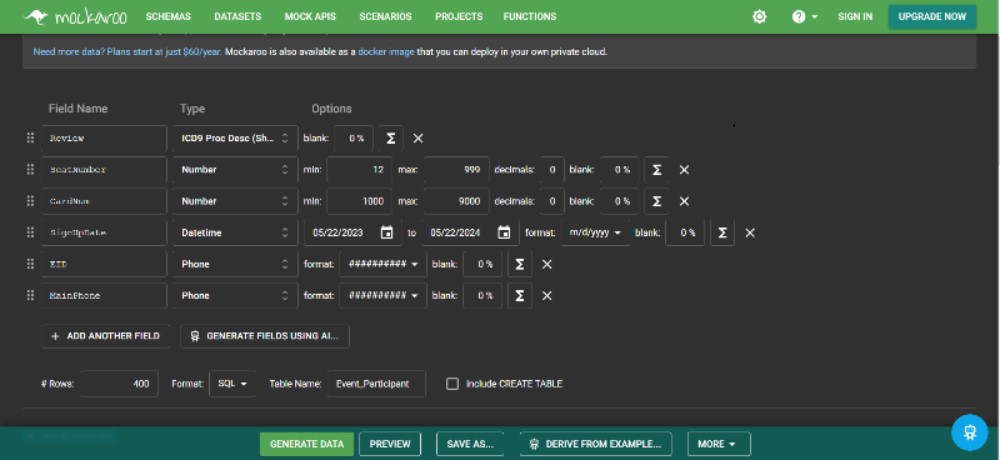


חלק מן הפקודות שנוצרו:

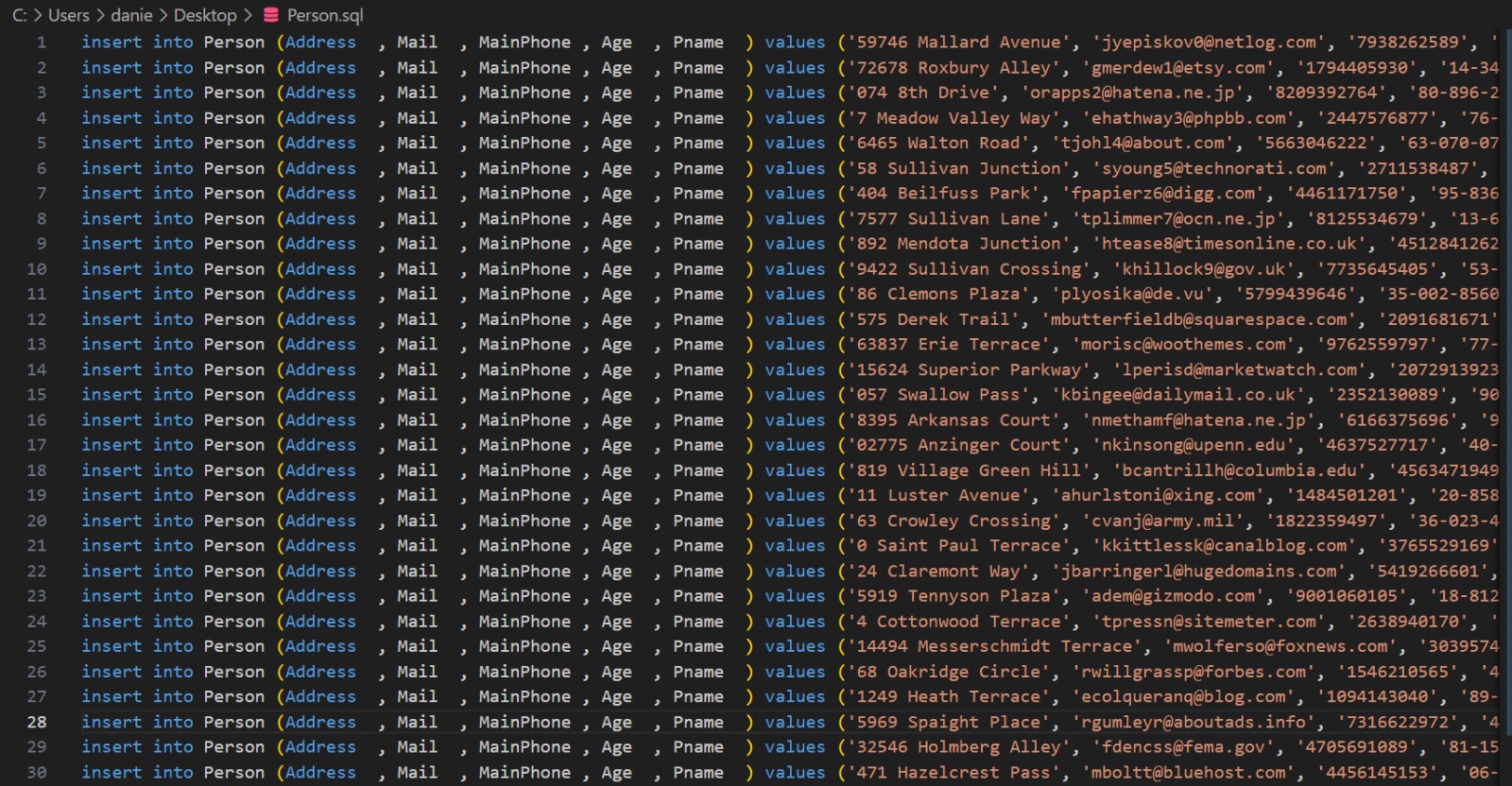




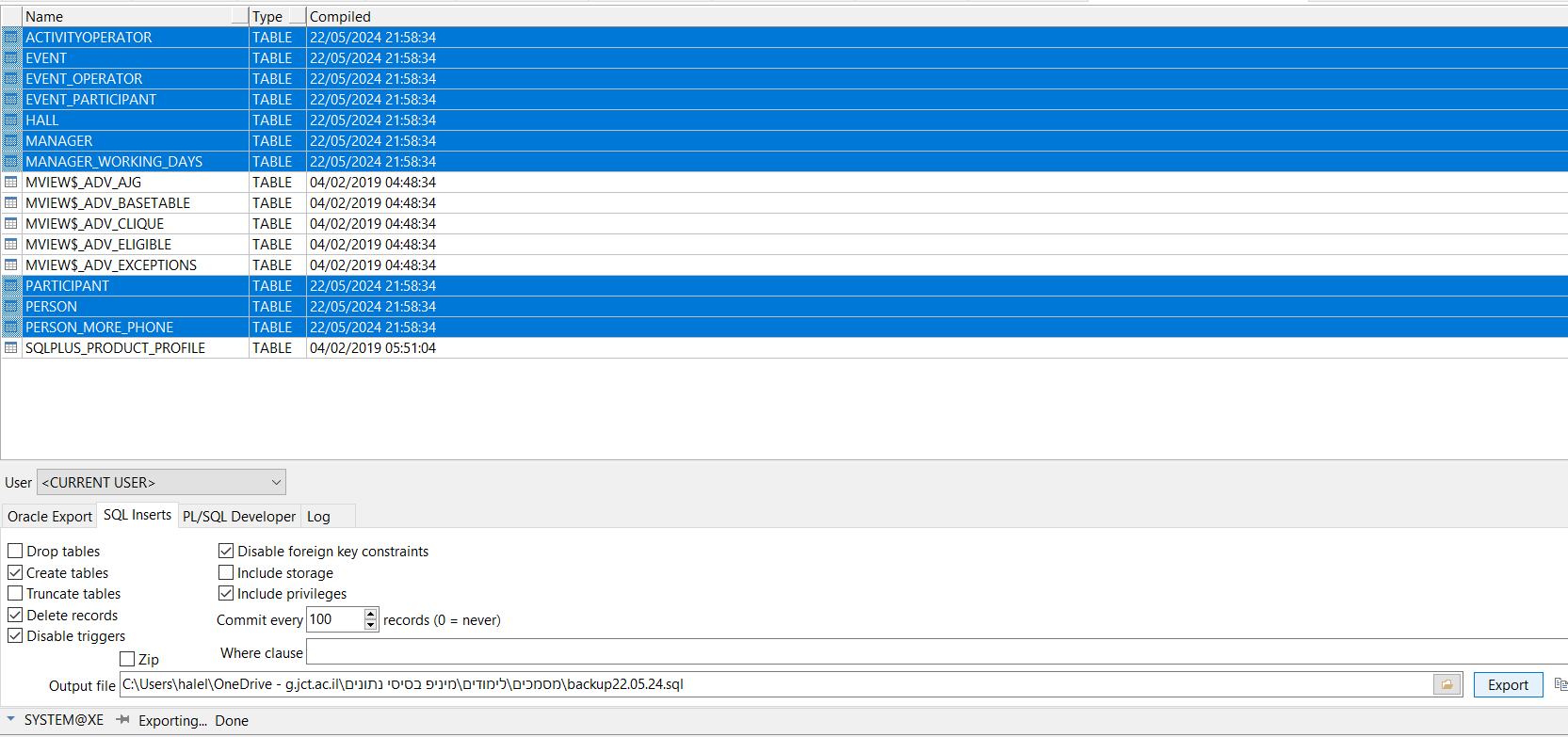
הכנסת נתונים דרך Mockaroo

שימוש באתר:

חלק מן הפקודות שנוצרו:



גיבוי ושחזור הטבלאות

גיבוי:

שחזור:

**שלב ב' – שאילתות ואילוצים**

**שאילתות select:**

1. שאילתא שמחזירה את רשימת המנהלים ה"מוצלחים" ביותר. כלומר, כל המנהלים שממוצע המשובים שנתנו משתתפים שנרשמו לפעילויות שניהלו גבוה מ 6.

הרשימה תכיל את השם, הטלפון וממוצע המשובים של כל מנהל, ותמוין מהמוצלח ביותר כלפי מטה.

SELECT

p.Pname AS ManagerName,

m.MainPhone AS ManagerPhone,

ROUND(AVG(ep.Review), 2) AS AverageReview

FROM

Manager m

JOIN

Person p ON m.MainPhone = p.MainPhone

JOIN

Event e ON m.MainPhone = e.MainPhone

JOIN

Event\_Participant ep ON e.EID = ep.EID

GROUP BY

p.Pname, m.MainPhone

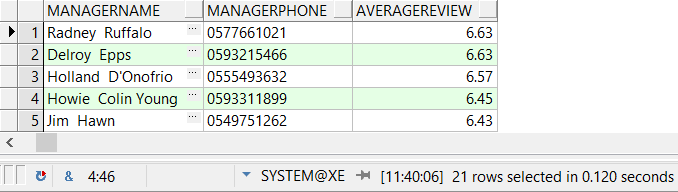
HAVING

ROUND(AVG(ep.Review), 2) > 6

ORDER BY

AverageReview DESC;

הרצה ותוצאות השאילתא:



1. שאילתא שמחזירה מידע על אירועים שיש בהם בעיית מקום. כלומר שמספר הרשומים אליהם גבוה מתכולת האולם בו הם משובצים.

עבור כל אירוע כזה נקבל את מס' האירוע, מס' האולם, מספר הנרשמים, ותכולת האולם.

SELECT

e.EID AS EventNumber,

e.HID AS HallNumber,

h.Capacity,

COUNT(ep.MainPhone) AS NumberOfRegistrants

FROM

Event e

JOIN

Hall h ON e.HID = h.HID

JOIN

Event\_Participant ep ON e.EID = ep.EID

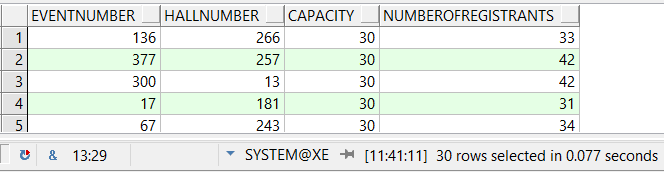
GROUP BY

e.EID, e.HID, h.Capacity;

HAVING

COUNT(ep.MainPhone) > h.Capacity;

הרצה ותוצאות השאילתא:



1. שאילתא שמחזירה מידע אודות "הרשמה מוקדמת". כלומר, רישומים לאירועים שהתבצעו לפחות שבוע מראש מתאריך האירוע.

נקבל את שם המשתתף, הטלפון שלו, מספר האירוע, תאריך האירוע, ותאריך הרישום.

SELECT

p.Pname AS ParticipantName,

ep.MainPhone,

e.EID AS EventNumber,

e.edate,

ep.signupdate

FROM

Event\_Participant ep

JOIN

Participant par ON ep.MainPhone = par.MainPhone

JOIN

Person p ON par.MainPhone = p.MainPhone

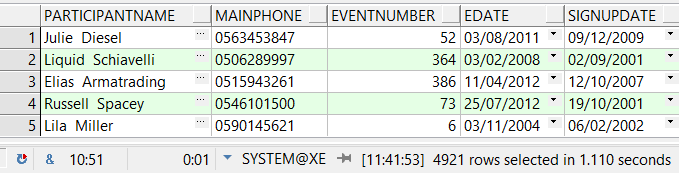
JOIN

Event e ON ep.EID = e.EID

WHERE

ep.SignUpDate <= e.EDate - 7;

הרצה ותוצאות השאילתא:



1. שאילתא שמחזירה את סוג הפעילות המועדפת על כל קבוצת גיל. כלומר, עבור כל קבוצת גיל את סוג הפעילות שיש הכי הרבה הרשמות לפעילויות מסוג זה.

נקבל רשימה של קבוצות הגיל, ממויינות בסדר עולה, והקטגוריה המועדפת לכל קבוצה.

WITH AgeGroups AS (

SELECT

CASE

WHEN Age BETWEEN 7 AND 10 THEN 'Mini-Club'

WHEN Age BETWEEN 11 AND 14 THEN 'Kids-Club'

WHEN Age BETWEEN 15 AND 18 THEN 'Teen-Club'

WHEN Age BETWEEN 19 AND 60 THEN 'Adults-Club'

WHEN Age BETWEEN 61 AND 90 THEN 'Elders-Club'

ELSE 'Unknown'

END AS AgeGroup,

MainPhone

FROM

Person

)

SELECT

AgeGroup,

(

SELECT

e.ECategory

FROM

Event e

JOIN

Event\_Participant ep ON e.EID = ep.EID

JOIN

AgeGroups ag ON ep.MainPhone = ag.MainPhone

WHERE

ag.AgeGroup = ag\_outer.AgeGroup

GROUP BY

e.ECategory

ORDER BY

COUNT(\*) DESC

FETCH FIRST 1 ROW ONLY

) AS MostPopularActivity

FROM

AgeGroups ag\_outer

GROUP BY

AgeGroup

ORDER BY

CASE

WHEN AgeGroup = 'Mini-Club' THEN 1

WHEN AgeGroup = 'Kids-Club' THEN 2

WHEN AgeGroup = 'Teen-Club' THEN 3

WHEN AgeGroup = 'Adults-Club' THEN 4

WHEN AgeGroup = 'Elders-Club' THEN 5

ELSE 6

END;

הרצה ותוצאות השאילתא: תמונה שמכילה טקסט, צילום מסך, תצוגה, גופן

התיאור נוצר באופן אוטומטי

**שאילתות delete:**

1. שאילתא שמוחקת אירועים שנקבעו ליום שאינו יום עבודה של המנהל האחראי עליהם. בהתאם נמחקים כל הרישומים לאירוע זה, וכן הקשר עם המפעיל שהוזמן לאירוע זה.

DELETE FROM Event\_Participant ep

WHERE ep.EID IN (

SELECT e.EID

FROM Event e

WHERE TO\_CHAR(e.EDate, 'Day') NOT IN (

SELECT mw.Working\_Days

FROM Manager\_Working\_Days mw

WHERE mw.MainPhone = e.MainPhone

)

);

DELETE FROM Event\_Operator eo

WHERE eo.EID IN (

SELECT e.EID

FROM Event e

WHERE TO\_CHAR(e.EDate, 'Day') NOT IN (

SELECT mw.Working\_Days

FROM Manager\_Working\_Days mw

WHERE mw.MainPhone = e.MainPhone

)

);

DELETE FROM Event e

WHERE TO\_CHAR(e.EDate, 'Day') NOT IN (

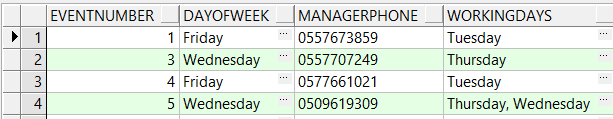
SELECT mw.Working\_Days

FROM Manager\_Working\_Days mw

WHERE mw.MainPhone = e.MainPhone

);

לפני ההרצה, נוכל לראות אירועים שהיום שנקבע להם אינו מתוך ימי העבודה של המנהל:



הרצה:



לאחר ההרצה, נותרו רק אירועים שהם באחד מימי העבודה של המנהל שלהם:



2. הספרייה החליטה שהתנדבות אינה סוג פעילות מתאים לילדים מתחת לגיל עשר. השאילתא להלן מוחקת את השתתפויותיהם של ילדים מתחת לגיל 10 מאירועים שבקטגוריה "התנדבות".

DELETE FROM

Event\_Participant

WHERE

EID IN (

SELECT e.EID

FROM Event e

WHERE e.ECategory = 'volunteering'

)

AND

MainPhone IN (

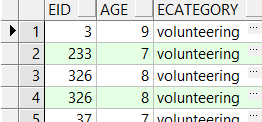
SELECT p.MainPhone

FROM Person p

WHERE p.Age < 10

);

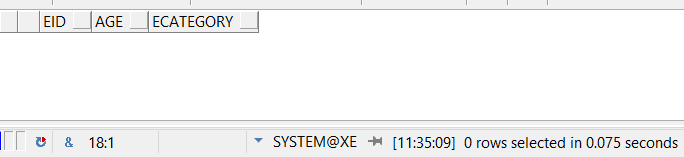
לפני ההרצה ניתן למצוא רישומים של ילדים עד גיל 10, שרשומים לאירוע מסוג התנדבות:



נריץ:



לאחר ההרצה לא נמצאו ילדים עד גיל 10 שרשומים להתנדבות:



**שאילתות update:**

1. שאילתא שמעדכנת את המחיר של אירועים שרוצים להגדיל את הרישום עליהם.  
מפחיתה ב 20% את מחיר הרישום לכל אירוע שבו מספר הנרשמים עד כה קטן מ50, וכן נותר עוד מקום באולם בו מתקיים האירוע.

*-- Update the price of events with fewer than 50 participants and available space in the hall*

UPDATE Event e

SET e.Price = CEIL(e.Price \* 0.80)

WHERE e.EID IN (

SELECT e.EID

FROM Event e

JOIN Hall h ON e.HID = h.HID

LEFT JOIN (

SELECT ep.EID, COUNT(ep.MainPhone) AS ParticipantCount

FROM Event\_Participant ep

GROUP BY ep.EID

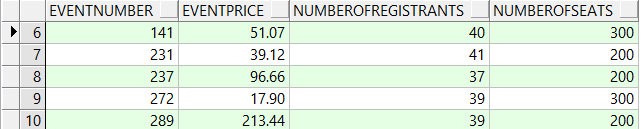
) epc ON e.EID = epc.EID

WHERE epc.ParticipantCount < 50

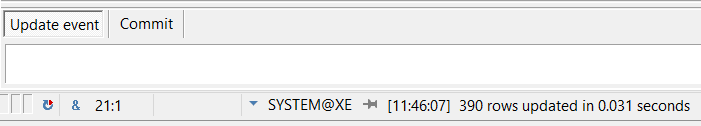
AND h.Capacity - COALESCE(epc.ParticipantCount, 0) > 0

);

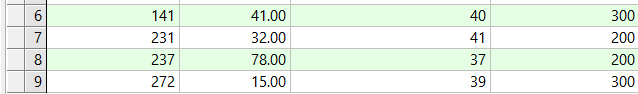
נראה את מחירם של אירועים מיועדים להנחה לפני ההרצה.  
ניתן לראות שמספר הנרשמים קטן מ50, ושיש עוד מקום באולם:



הרצה:



לאחר ההרצה- ניתן לראות שהמחיר לכרטיס עבור אירועים אלו הוזל ב 20%:



2. שאילתא שמעדכנת אירועים ששובצו להתקיים בקומה גבוהה (קומה 2 ומעלה) וקהל היעד שלהם הוא גיל הזהב, להיות מיועדים למבוגרים ולא לגיל הזהב. מכיוון שהעלייה במדרגות לא מתאימה לקהל יעד זה.

UPDATE Event e

SET e.Audience = 'Adults'

WHERE e.Audience = 'Elderly'

AND e.HID IN (

SELECT h.HID

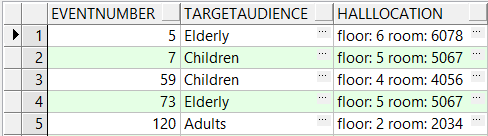
FROM Hall h

WHERE REGEXP\_SUBSTR(h.Place, 'floor: ([0-9]+)') IS NOT NULL

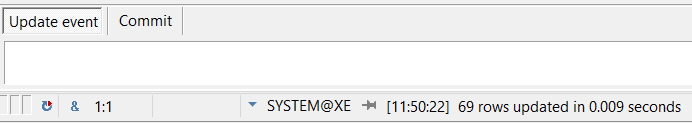
AND TO\_NUMBER(REGEXP\_SUBSTR(h.Place, 'floor: ([0-9]+)', 1, 1, NULL, 1)) >= 2

);

לפני ההרצה- ישנם אירועים שמתקיימים בקומה גבוהה ומיודעם לגיל הזהב:



הרצה:



לאחר ההרצה:

תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

**שאילתות עם פרמטרים:**

1. השאילתא מחזירה את כל המנהלים שממוצע הדירוגים של אירועים שניהלו גבוהים מהמספר המבוקש.  
המשתמש יכניס דירוג (בין 1 ל 10), ויקבל את רשימת כל המנהלים שממוצע הדירוג של אירועים שניהלו גבוה מציון זה. עבור כל מנהל יוחזר שמו, מס' הפלאפון שלו, והממוצע.  
אם המשתמש לא בוחר אחרת, ברירת המחדל היא 6.

*--Query 1: hight rating managers*

SELECT

p.Pname AS ManagerName,

m.MainPhone AS ManagerID,

ROUND(AVG(ep.Review), 2) AS AverageReview

FROM

Manager m

JOIN

Person p ON m.MainPhone = p.MainPhone

JOIN

Event e ON m.MainPhone = e.MainPhone

JOIN

Event\_Participant ep ON e.EID = ep.EID

GROUP BY

p.Pname, m.MainPhone

HAVING

ROUND(AVG(ep.Review), 2) > &<name="min rating" hint="min rating between 3-10" type="integer" default=6>

ORDER BY

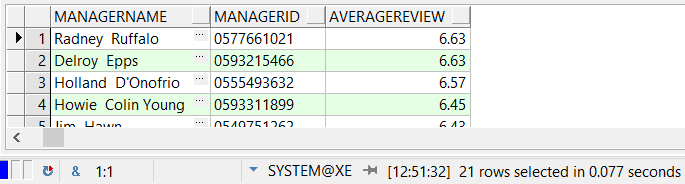
AverageReview DESC;

הרצה:

תמונה שמכילה טקסט, צילום מסך, קו, גופן

התיאור נוצר באופן אוטומטי

תוצאות השאילתא (עבור 6):



2. השאילתא מחזירה את פרטי המשתתפים מבין טווח גילאים רצוי, שמשתתפים בפעילויות מהקטגוריות המבוקשות.  
המשתמש בוחר קטגוריות (ניתן לבחור יותר מאחת), בוחר גיל מינימלי וגיל מקסימלי (ברירת המחדל היא 10 ו50), ומקבל את פרטי הרישומים של אנשים מטווח גילאים זה לאירועים מקטגוריות אלו.

*--Query 2: paticipant in range of ages categories*

select \* FROM Event\_Participant

WHERE

EID IN (

SELECT e.EID

FROM Event e

WHERE e.ECategory = &<multiselect="yes" name="Event category" list="'Lecture', 'workshop', 'performance', 'story', 'educational activity', 'play', 'conversation with an author', 'educational activity', 'training', 'film screening', 'exhibition', 'sports', 'volunteering" required=true >

)

AND

MainPhone IN (

SELECT p.MainPhone

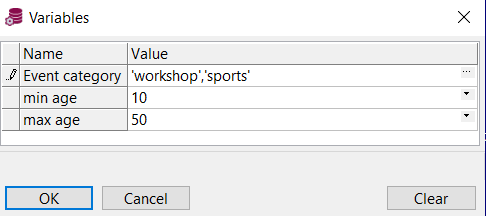
FROM Person p

WHERE p.Age > &<name="min age" type="integer" default=10> and p.Age < &<name="max age" type="integer" default=50>

);

תמונה שמכילה טקסט, צילום מסך, תוכנה, תצוגה

התיאור נוצר באופן אוטומטיהרצה:



תוצאות: כל הרישומים של משתתפים מגיל 10 עד 50 לאירועים מסוג ספורט ומסוג סדנא-

תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

3. השאילתא מחזירה רשימה של רישומים מוקדמים לאירועים.

המשתמש מכניס מספר ימים, נניח X, ומקבל את הפרטים של כל הרישומים (שם ומס' טלפון של המשתתף, מס' האירוע, תאריך הרישום ותאריך האירוע) שהתבצעו X ימים מראש.  
ברירת המחדל היא 7 ימים.

*--Query 3: early sign up*

SELECT

p.Pname AS ParticipantName,

ep.MainPhone,

e.EID AS EventNumber,

e.edate,

ep.signupdate

FROM

Event\_Participant ep

JOIN

Participant par ON ep.MainPhone = par.MainPhone

JOIN

Person p ON par.MainPhone = p.MainPhone

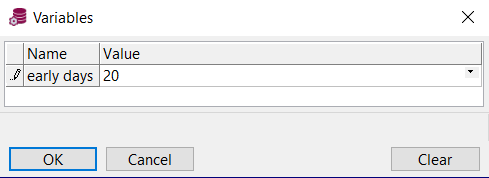
JOIN

Event e ON ep.EID = e.EID

WHERE

ep.SignUpDate <= e.EDate - &<name="early days" type=integer default=7>;

הרצה:



ביקשנו הרשמות של 20 ימים מראש.

תוצאות:

תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

4. השאילתא מבקשת מהמשתמש טווח תאריכים (תאריך התחלה ותאריך סיום), ומחזירה את כל המפעילים שהוזמנו בתקופה זו ולא היה כדאי להזמין אותם: כל המפעילים שמחירם גבוה מ 2000 ₪, ובפעילויות שהעבירו בתקופה הזו השתתפו בממוצע פחות מ 50 איש.  
עבור כל מפעיל כזה נקבל את מס' הפלאפון שלו, עלותו, ומס' המשתתפים הממוצע בפעילויות שלו בתקופה זו.

*--Query 4: Operators that should not have been ordered, in a specific period*

SELECT AO.MainPhone, AO.OCost, round(AVG(EP.ParticipantCount),0) AS AvgParticipants

FROM ActivityOperator AO

JOIN Event\_Operator EO ON AO.MainPhone = EO.MainPhone

JOIN Event E ON EO.EID = E.EID

JOIN (

SELECT EID, COUNT(MainPhone) AS ParticipantCount

FROM Event\_Participant

GROUP BY EID

) EP ON E.EID = EP.EID

WHERE E.EDate BETWEEN &<name="d\_from" type="date" default="01/01/24"> and &<name="d\_to" type="date" default="01/09/24">

AND AO.OCost > 2000

GROUP BY AO.MainPhone, AO.OCost

HAVING AVG(EP.ParticipantCount) < 50;

הרצה:

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי

תוצאות (עבור ערכי ברירת המחדל, שבצילום לעיל):

תמונה שמכילה טקסט, גופן, קו, מספר

התיאור נוצר באופן אוטומטי

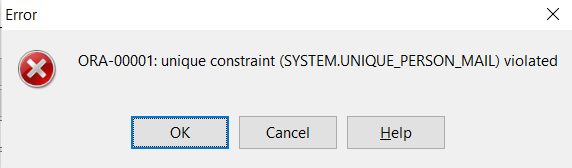
**אילוצים:**

1. הוספת אילוץ unique עבור כתובת אימייל בטבלה Person.  
כלומר, לא יתאפשר להכניס כתובת אימייל שכבר שמורה עבור אדם אחר.

ALTER TABLE Person

ADD CONSTRAINT unique\_person\_mail UNIQUE (Mail);

ננסה להכניס עבור שני אנשים את אותה כתובת אימייל, וקיבלנו את השגיאה הבאה:

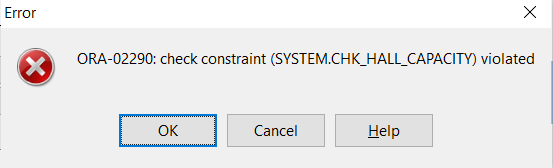


2. הוספת אילוץ check לתכונת ה capacity בטבלה Hall. כך שקיבולת האולם חייבת להיות גדולה מ 0 אנשים.

ALTER TABLE Hall

ADD CONSTRAINT chk\_hall\_capacity CHECK (Capacity > 0);

ננסה להכניס אולם עם קיבולת 0 ונקבל שגיאה:

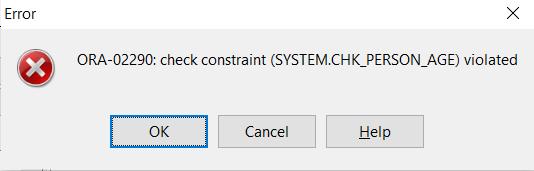


3. הוספת אילוץ check לתכונה age בטבלה Person,  
כך שהגיל של אדם חייב להיות מעל 0.

ALTER TABLE Person

ADD CONSTRAINT chk\_person\_age CHECK (Age > 0);

ננסה להכניס אדם שגילו 0 ונקבל שגיאה:

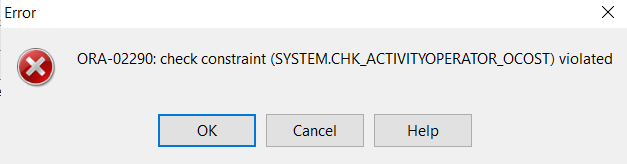


4. הוספת אילוץ check לתכונה cost בטבלה operator, כך שעלות של מפעיל לא יכולה להיות מספר שלילי.

ALTER TABLE ActivityOperator

ADD CONSTRAINT chk\_activityoperator\_ocost CHECK (OCost >= 0);

ננסה להכניס מפעיל שעלותו 1- ונקבל שגיאה:

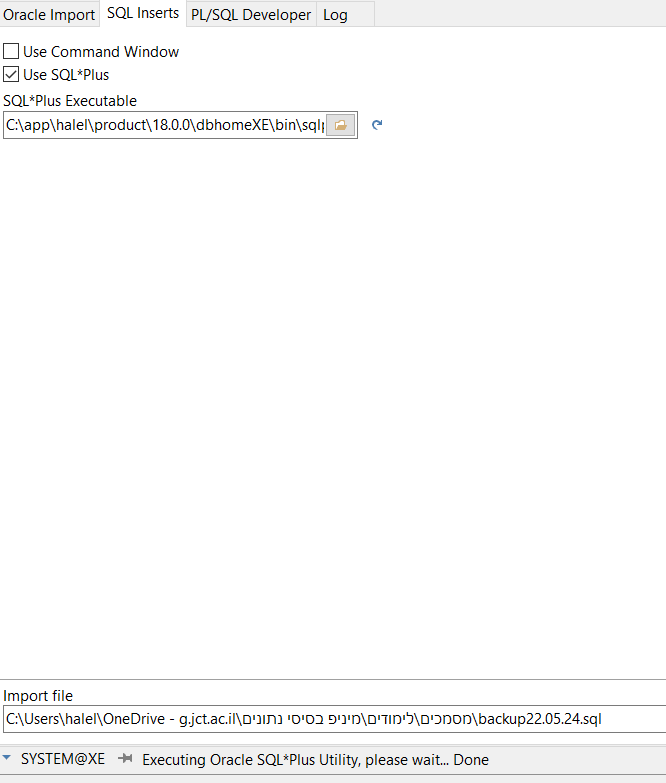
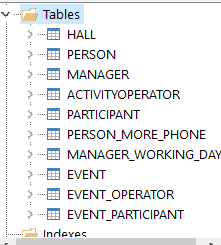


5. הוספת default עבור התכונה price בטבלה event, כך שמחיר ברירת מחדל של אירוע יהיה 50 ₪.

נכניס אולם חדש בלי לשלוח ערך עבור עלות האירוע, ונראה שנכנס הערך 50 בתכונה PRICE:

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי



**שלב ג' – תכניות**

**תכנית א'-** **updateOperatorPrace:**

התכנית מבצעת שינויים באירועים עתידיים לפי אימות היותו של מפעיל מסוים "מפעיל ברמה גבוהה"  
התכנית מקבלת מהמשתמש מספר פלאפון של מפעיל (אם המספר אינו של מפעיל, תיזרק שגיאה מתאימה).  
ראשית, היא בודקת האם המפעיל הנ"ל הוא מפעיל מוצלח, על ידי קריאה לפונקציה DeservesRaiseOperator. הפונקציה בודקת האם ממוצע כל המשובים שנתנו משתתפים עבור אירועים שהעביר מפעיל זה הינו 6 ומעלה.  
אם כן, הפונקציה תחזיר אמת. אחרת- תחזיר שקר.

לאחר מכן התכנית תבצע הסתעפות על פי הערך הבוליאני המוחזר. אם חזר שקר (המפעיל לא נחשב "ברמה גבוהה") – יודפס שהמפעיל אינו עונה להגדרה, והתכנית תסתיים.  
אם יוחזר אמת – תיקרא הפרוצדורה ManageOperatorEvents. הפרוצדורה תקבל את מספר המפעיל ותבצע מספר שינויים בנתונים:

1. תמצא מבין האירועים העתידיים שמועברים על ידי מפעיל זה את שלושת האירועים שאליהם רשום המספר הגבוה ביותר של משתתפים, ותעלה את המחיר לכרטיס ב 20%.
2. תמצא את המפעיל הגרוע ביותר (זה שממוצע הדירוגים שנתנו המשתתפים באירועים שהעביר הוא הנמוך ביותר), ותחליף אותו במפעיל הנ"ל (שנמצא כי הוא "ברמה גבוהה"). כלומר, עבור כל אירוע עתידי של המפעיל הגרוע – יוחלף המפעיל למפעיל שלנו, באם הדבר מתאפשר ואין התנגשות של תאריכים.

קוד התכנית:

DECLARE

*--v\_OperatorPhone ActivityOperator.MainPhone%TYPE := '&operator\_phone';*

v\_DeservesRaise BOOLEAN;

BEGIN

*-- Call the function*

v\_DeservesRaise := DeservesRaiseOperator(p\_OperatorPhone=> :p\_OperatorPhone);

IF v\_DeservesRaise THEN

*-- Call the procedure*

DBMS\_OUTPUT.PUT\_LINE('The operator deserves to raise' || SQLERRM);

ManageOperatorEvents(p\_OperatorPhone=> :p\_OperatorPhone);

else

DBMS\_OUTPUT.PUT\_LINE('The operator does not deserves to raise' || SQLERRM);

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

קוד הפונקציה DeservesRaiseOperator:

CREATE OR REPLACE FUNCTION DeservesRaiseOperator(p\_OperatorPhone IN VARCHAR2)

RETURN BOOLEAN

AS

v\_AvgRating NUMBER := 0;

v\_PhoneExists NUMBER;

e\_OperatorNotFound EXCEPTION;

BEGIN

*-- Check if the phone number belongs to an operator*

SELECT COUNT(\*)

INTO v\_PhoneExists

FROM ActivityOperator

WHERE MainPhone = p\_OperatorPhone;

IF v\_PhoneExists = 0 THEN

RAISE e\_OperatorNotFound; *-- Raise custom exception*

END IF;

*-- Calculate the average rating using the CalculateAverageRating function*

v\_AvgRating := CalculateAverageRating(p\_OperatorPhone);

RETURN v\_AvgRating > 5;

EXCEPTION

WHEN e\_OperatorNotFound THEN

RAISE\_APPLICATION\_ERROR(-20001, 'The cell phone number entered does not belong to an operator');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20012, 'An error occurred while calculating the average rating: ' || SQLERRM);

END DeservesRaiseOperator;

קוד הפרוצדורה ManageOperatorEvents:

CREATE OR REPLACE PROCEDURE ManageOperatorEvents(

p\_OperatorPhone IN VARCHAR2

)

AS

CURSOR upcoming\_events IS

SELECT e.EID, e.Price

FROM Event e

WHERE e.OpMainPhone = p\_OperatorPhone

AND e.EDate > SYSDATE

ORDER BY (SELECT COUNT(\*) FROM Event\_Participant ep WHERE ep.EID = e.EID) DESC

FETCH FIRST 3 ROWS ONLY;

CURSOR worst\_operator IS

SELECT ao.MainPhone

FROM ActivityOperator ao;

v\_EID Event.EID%TYPE;

v\_Price Event.Price%TYPE;

v\_WorstOperatorPhone ActivityOperator.MainPhone%TYPE;

v\_EventConflict NUMBER;

v\_EventsUpdated NUMBER := 0;

v\_EventsReassigned NUMBER := 0;

v\_TotalEvents NUMBER := 0;

v\_LowestAvgRating NUMBER := 1000;

v\_CurrentAvgRating NUMBER;

BEGIN

*-- Debug print: Start of the procedure*

DBMS\_OUTPUT.PUT\_LINE('Starting ManageOperatorEvents procedure for operator phone: ' || p\_OperatorPhone);

*-- Count the total number of upcoming events for the operator*

SELECT COUNT(\*)

INTO v\_TotalEvents

FROM Event e

WHERE e.OpMainPhone = p\_OperatorPhone

AND e.EDate > SYSDATE;

DBMS\_OUTPUT.PUT\_LINE('The operator is responsible for ' || v\_TotalEvents || ' upcoming events.');

*-- Increase the price of the top 3 upcoming events by 20%*

FOR event\_rec IN upcoming\_events LOOP

v\_EID := event\_rec.EID;

v\_Price := event\_rec.Price \* 1.2;

UPDATE Event

SET Price = v\_Price

WHERE EID = v\_EID;

v\_EventsUpdated := v\_EventsUpdated + 1;

DBMS\_OUTPUT.PUT\_LINE('Updated price for event ID: ' || v\_EID || ', new price: ' || v\_Price);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('The price of ' || v\_EventsUpdated || ' events has been updated.');

*-- Find the worst operator by calculating average ratings*

OPEN worst\_operator;

LOOP

FETCH worst\_operator INTO v\_WorstOperatorPhone;

EXIT WHEN worst\_operator%NOTFOUND;

v\_CurrentAvgRating := CalculateAverageRating(v\_WorstOperatorPhone);

IF v\_CurrentAvgRating < v\_LowestAvgRating THEN

v\_LowestAvgRating := v\_CurrentAvgRating;

v\_WorstOperatorPhone := v\_WorstOperatorPhone;

END IF;

END LOOP;

CLOSE worst\_operator;

DBMS\_OUTPUT.PUT\_LINE('Worst operator phone: ' || v\_WorstOperatorPhone);

IF v\_WorstOperatorPhone IS NOT NULL THEN

*-- Reassign future events of the worst operator to the specified operator*

FOR event\_rec IN (

SELECT EID, EDate, ETime

FROM Event

WHERE OpMainPhone = v\_WorstOperatorPhone

AND EDate > SYSDATE

) LOOP

*-- Check for scheduling conflicts*

SELECT COUNT(\*)

INTO v\_EventConflict

FROM Event e

WHERE e.OpMainPhone = p\_OperatorPhone

AND e.EDate = event\_rec.EDate

AND e.ETime = event\_rec.ETime;

IF v\_EventConflict = 0 THEN

UPDATE Event

SET OpMainPhone = p\_OperatorPhone

WHERE EID = event\_rec.EID;

v\_EventsReassigned := v\_EventsReassigned + 1;

DBMS\_OUTPUT.PUT\_LINE('Reassigned event ID: ' || event\_rec.EID || ' to operator phone: ' || p\_OperatorPhone);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Event ID ' || event\_rec.EID || ' on date ' || event\_rec.EDate || ' at time ' || event\_rec.ETime || ' cannot be reassigned due to a scheduling conflict.');

END IF;

END LOOP;

IF v\_EventsReassigned > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('The operator with the lowest average reviews has been reassigned for ' || v\_EventsReassigned || ' events.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('No events were reassigned due to scheduling conflicts.');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('No operator found with the lowest average reviews.');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20013, 'An error occurred while managing operator events: ' || SQLERRM);

END ManageOperatorEvents;

קוד פונקציית עזר לחישוב ממוצע הדירוגים של מפעיל – CalculateAverageRating:

CREATE OR REPLACE FUNCTION CalculateAverageRating(p\_OperatorPhone IN VARCHAR2)

RETURN NUMBER

AS

v\_TotalRating NUMBER := 0;

v\_Count NUMBER := 0;

v\_AvgRating NUMBER := 0;

BEGIN

FOR rating\_rec IN (

SELECT ep.Review

FROM Event e

JOIN Event\_Participant ep ON e.EID = ep.EID

WHERE e.OpMainPhone = p\_OperatorPhone

)

LOOP

v\_TotalRating := v\_TotalRating + rating\_rec.Review;

v\_Count := v\_Count + 1;

END LOOP;

IF v\_Count > 0 THEN

v\_AvgRating := v\_TotalRating / v\_Count;

ELSE

v\_AvgRating := 0;

END IF;

RETURN v\_AvgRating;

EXCEPTION

WHEN OTHERS THEN

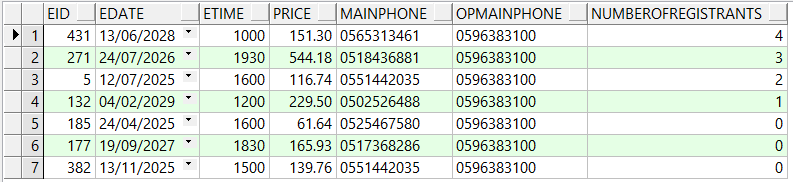
RAISE\_APPLICATION\_ERROR(-20014, 'An error occurred while calculating the average rating: ' || SQLERRM);

END CalculateAverageRating;

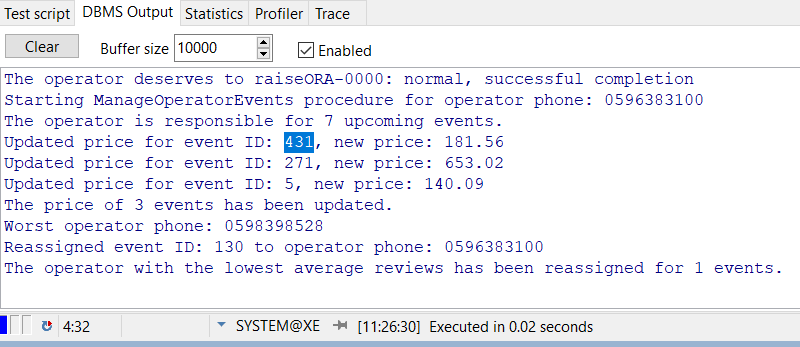
**דוגמאות הרצה:**

מקרה א': מפעיל שכן ענה להגדרה "מפעיל ברמה גבוהה" ובוצעו עבורו העדכונים:

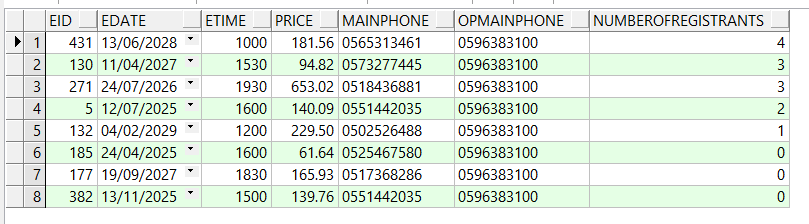
לפני ההרצה- אלו האירועים העתידיים שמפעיל המפעיל שמספרו 0596383100, ממויינים לפי מספר הנרשמים.



הרצה:

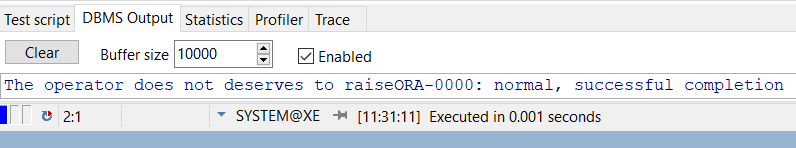


אחרי ההרצה- ניתן לראות שהמחיר של שלושת האירועים הראשונים השתנה, וכן שהתווסף אירוע נוסף (שהיה שייך למפעיל הגרוע ביותר):



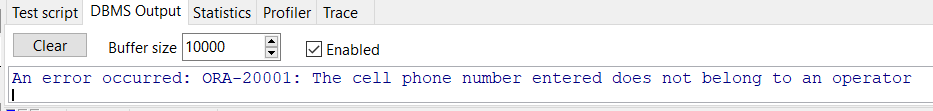
מקרה ב'- מפעיל שלא ענה להגדרה

תודפס הודעה מתאימה ולא יתבצעו שינויים בבסיס הנתונים:



מקרה ג'- מקרה של שגיאה – אם הוכנס מספר פלאפון שאינו של מפעיל

תודפס הודעה מתאימה, ולא יתבצעו שינויים בבסיס הנתונים (פונקציית העדכון כלל לא נקראת):



**תכנית ב'- hallsAndTrainingDay:**

הנהלת הספרייה מעוניינת לקבוע יום השתלמות למנהלים.

ראשית, התכנית קוראת לפונקציה FindSuitableTrainingDay שתמצא תאריך מתאים ואולם מתאים להשתלמות. הפונקציה תחפש את היום מימי השבוע שבו יש מינימום מנהלים שזהו יום עבודה שלהם. לאחר מכן, תמצא אולם שבו אפשר לערוך את ההשתלמות בהקדם ביום זה בשבוע, אשר התכולה שלו מתאימה להכיל את כלל המנהלים של הספרייה.  
הפונקציה מחזירה את התאריך והאולם שנבחרו עבור יום ההשתלמות, והתוכנית תדפיס את מה שהתקבל.

בנוסף, ההנהלה מעוניינת בבדיקת העדפות הקהל ביחס למגדר המפעילים בסוגי הפעילויות השונות. לשם כך, התכנית מזמנת את הפרוצדורה PrintPreferredOperatorType. הפרוצדורה בודקת עבור כל קטגוריית אירועים (למשל- סדנאות/ ספורט/ הצגה/ וכו') האם מספר הנרשמים הממוצע גבוה יותר כאשר המפעיל הוא גבר או כאשר זו מפעילה אישה.  
התוצאות מודפסות במסך הקלט.

אם לא נמצא אולם מתאים או התרחשה כל שגיאה אחרת – תיזרק שגיאה.

קוד התכנית:

DECLARE

v\_TrainingDetails VARCHAR2(3000); *-- Increase the size to handle larger return values*

BEGIN

*-- Call the function FindSuitableTrainingDay*

v\_TrainingDetails := FindSuitableTrainingDay;

DBMS\_OUTPUT.PUT\_LINE('Training Details: ' || v\_TrainingDetails);

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('');

*-- Call the procedure PrintPreferredOperatorType*

PrintPreferredOperatorType();

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

קוד הפונקציה FindSuitableTrainingDay:

CREATE OR REPLACE FUNCTION FindSuitableTrainingDay

RETURN VARCHAR2

IS

v\_min\_managers\_count INT := 900;

v\_min\_managers\_day VARCHAR2(10);

v\_managers\_count INT;

v\_training\_date DATE;

v\_hall\_id INT;

v\_capacity INT;

v\_manager\_count INT;

v\_event\_count INT;

CURSOR c\_days IS

SELECT DISTINCT Working\_Days

FROM Manager\_Working\_Days;

CURSOR c\_managers(day IN VARCHAR2) IS

SELECT COUNT(\*) AS manager\_count

FROM Manager\_Working\_Days

WHERE Working\_Days = day;

CURSOR c\_halls(capacity\_needed IN INT) IS

SELECT HID, Capacity

FROM Hall

WHERE Capacity >= capacity\_needed

ORDER BY Capacity;

CURSOR c\_events(hall\_id IN INT, training\_date IN DATE) IS

SELECT COUNT(\*) AS event\_count

FROM Event

WHERE HID = hall\_id AND EDate = training\_date; *-- Ensure EDate is the correct column name*

BEGIN

*-- Find the day of the week with the minimum number of managers working*

FOR day\_rec IN c\_days LOOP

OPEN c\_managers(day\_rec.Working\_Days);

FETCH c\_managers INTO v\_managers\_count;

CLOSE c\_managers;

IF v\_managers\_count < v\_min\_managers\_count THEN

v\_min\_managers\_count := v\_managers\_count;

v\_min\_managers\_day := day\_rec.Working\_Days;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Minimum managers working day: ' || v\_min\_managers\_day);

*-- Find the nearest date starting from today which is the desired day of the week.*

v\_training\_date := NEXT\_DAY(SYSDATE, v\_min\_managers\_day);

DBMS\_OUTPUT.PUT\_LINE('Closest training date: ' || TO\_CHAR(v\_training\_date, 'DD-MON-YYYY'));

*-- Find an available hall with sufficient capacity and no event on the training date*

SELECT COUNT(\*) INTO v\_manager\_count FROM Manager;

OPEN c\_halls(v\_manager\_count);

LOOP

FETCH c\_halls INTO v\_hall\_id, v\_capacity;

EXIT WHEN c\_halls%NOTFOUND;

*-- Check if there is any event scheduled in the hall on the training date*

OPEN c\_events(v\_hall\_id, v\_training\_date);

FETCH c\_events INTO v\_event\_count;

CLOSE c\_events;

*-- If no event is scheduled, select this hall*

IF v\_event\_count = 0 THEN

EXIT;

END IF;

END LOOP;

CLOSE c\_halls;

IF v\_event\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Selected hall ID: ' || v\_hall\_id || ' with capacity: ' || v\_capacity);

RETURN 'Training Date: ' || TO\_CHAR(v\_training\_date, 'DD-MON-YYYY') || ', Hall ID: ' || v\_hall\_id;

ELSE

RETURN 'No suitable hall found.';

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No suitable hall found.');

RETURN 'No suitable hall found.';

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

RETURN 'An unexpected error occurred.';

END FindSuitableTrainingDay;

קוד הפרוצדורה PrintPreferredOperatorType:

CREATE OR REPLACE PROCEDURE PrintPreferredOperatorType

IS

v\_avg\_registrants\_men NUMBER;

v\_avg\_registrants\_women NUMBER;

*-- Cursor to get the categories.*

CURSOR c\_categories IS

SELECT DISTINCT ECategory

FROM Event;

*-- Cursor to get the average number of registrants for events by male operators*

CURSOR c\_avg\_registrants\_men(category VARCHAR2) IS

SELECT AVG(participant\_count)

FROM (

SELECT e.EID, COUNT(p.mainPhone) AS participant\_count

FROM Event e

JOIN Event\_Participant p ON e.EID = p.EID

JOIN ActivityOperator o ON e.OpMainPhone = o.MainPhone

WHERE e.ECategory = category AND o.Gender = 'male'

GROUP BY e.EID

);

*-- Cursor to get the average number of registrants for events by female operators*

CURSOR c\_avg\_registrants\_women(category VARCHAR2) IS

SELECT AVG(participant\_count)

FROM (

SELECT e.EID, COUNT(p.mainPhone) AS participant\_count

FROM Event e

JOIN Event\_Participant p ON e.EID = p.EID

JOIN ActivityOperator o ON e.OpMainPhone = o.MainPhone

WHERE e.ECategory = category AND o.Gender = 'female'

GROUP BY e.EID

);

BEGIN

*-- Loop through each category*

FOR category\_rec IN c\_categories LOOP

*-- Get the average number of registrants for male operators*

OPEN c\_avg\_registrants\_men(category\_rec.ECategory);

FETCH c\_avg\_registrants\_men INTO v\_avg\_registrants\_men;

CLOSE c\_avg\_registrants\_men;

*-- Get the average number of registrants for female operators*

OPEN c\_avg\_registrants\_women(category\_rec.ECategory);

FETCH c\_avg\_registrants\_women INTO v\_avg\_registrants\_women;

CLOSE c\_avg\_registrants\_women;

*-- Print the preferred operator type for the category*

IF v\_avg\_registrants\_men > v\_avg\_registrants\_women THEN

DBMS\_OUTPUT.PUT\_LINE(category\_rec.ECategory || ': Male');

ELSE

DBMS\_OUTPUT.PUT\_LINE(category\_rec.ECategory || ': Female');

END IF;

END LOOP;

EXCEPTION

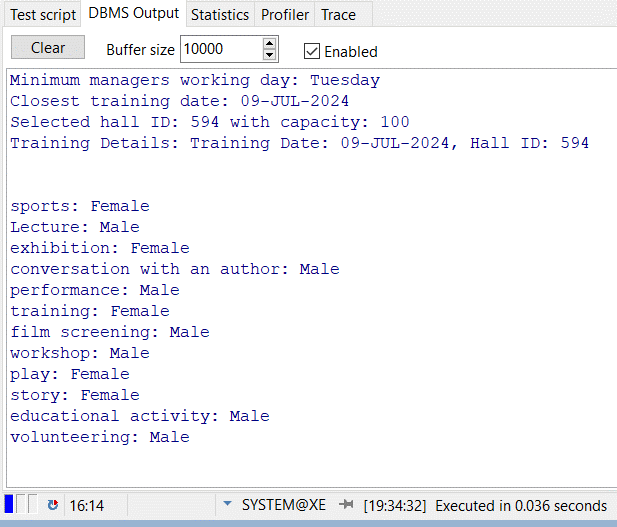
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END PrintPreferredOperatorType;

**דוגמת הרצה:**

ניתן לראות שהיום בשבוע שנבחר עבור ההשתלמות הוא יום שלישי, והתאריך שנקבע הוא ליום שלישי הקרוב (9.7.2024), באולם מספר 594.  
וכן ניתן לראות את הרשימה שמציגה עבור כל קטגוריית פעילות, איזה מגדר מפעיל מועדף יותר על הקהל.



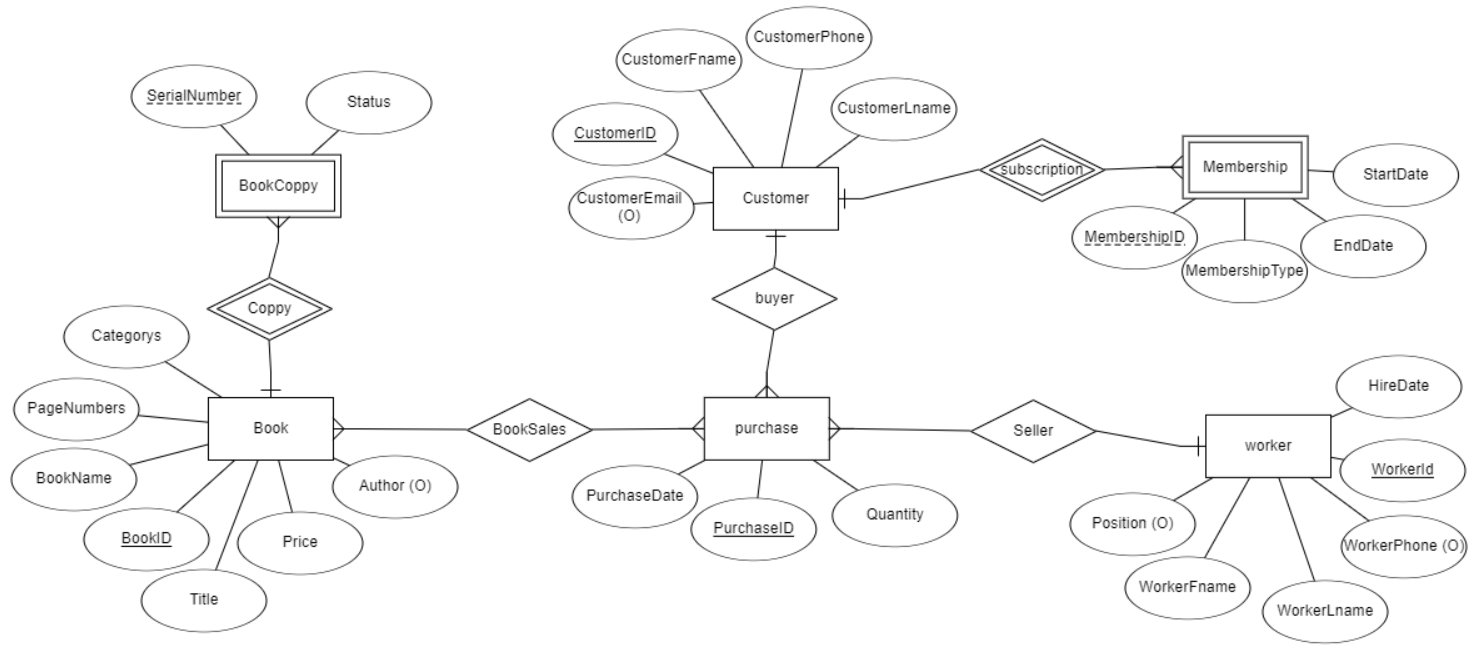
**- שלב ד' –**

**DSD של האגף החדש:**

תמונה שמכילה טקסט, צילום מסך, מספר, מקביל

התיאור נוצר באופן אוטומטי

**ERD של האגף החדש:**



**החלטות אינטגרציה:**

הטבלאות החדשות:

* נוסיף לטבלאות האגף שלנו את כל הטבלאות של האגף השני. ונבצע את השינויים הבאים:
* את שם הטבלה "Worker" המקורית, נשנה ל "Librarian", משום שבאיחוד האגפים יש יותר מסוג עובד אחד, ולכן עלינו להיות ספציפיים יותר.
* ניצור טבלה אחרת בשם "Worker" שתייצג "עובד" מכל סוג שהוא.

קשרים בין טבלאות:

* Librarian ו Manager ירשו מWorker משום שאלו סוגים של עובדים.
* Worker ירש מ Person משום שמדובר באנשים.
* Customer ירש מ Person, משום שמדובר באנשים.
* נקבע את MemberShip להיות ישות חלשה מ Person

שינויים בתכונות:

* נוסיף ל worker את התכונה hireDate שתחליף את ה hireDate של Librarian וה seniority של Manager (נחשב את תאריך הגיוס לפי שנות הותק בהתאם לתאריך הנוכחי).
* מכל הטבלאות שכעת יורשות מ Person נסיר תכונות של פרטים אישיים שנמצאות ב Person, ונקבע את התכונה mainPhone להיות המפתח.
* נסיר את התכונה "type" של participant שציינה האם משתתף הוא מנוי או לא. משום שיש לנו טבלה של membership שמציינת מנוי של אדם.
* בהתאם, נסיר את התכונות האופציונאליות של שם משתמש וסיסמא שהיו ב Participan.
* נסיר את התכונה position מ Librarian, משום שעכשיו יש פשוט שני סוגים של עובדים וקבענו שרק ספרן יכול לבצע מכירות.

**ERD של מיזוג האגפים:**

**תמונה שמכילה תרשים, ציור, שרטוט, דפוס

התיאור נוצר באופן אוטומטי**

תמונה שמכילה טקסט, צילום מסך, מקביל, מספר

התיאור נוצר באופן אוטומטי**DSD של מיזוג האגפים:**

**הסבר מילולי של התהליך והפקודות:**

מחיקת אילוצים לא רלוונטיים למבנה החדש:

SELECT constraint\_name

FROM all\_cons\_columns

WHERE table\_name = 'MEMBERSHIP' AND column\_name='CUSTOMERID';

ALTER TABLE Membership DROP CONSTRAINT SYS\_C0016027;

ALTER TABLE Membership DROP CONSTRAINT SYS\_C0016028;

ALTER TABLE Membership DROP CONSTRAINT SYS\_C0016029;

SELECT constraint\_name

FROM all\_cons\_columns

WHERE table\_name = 'PURCHASE' AND column\_name='WORKERID';

ALTER TABLE purchase DROP CONSTRAINT SYS\_C0016014;

ALTER TABLE purchase DROP CONSTRAINT SYS\_C0016017;

SELECT constraint\_name

FROM all\_cons\_columns

WHERE table\_name = 'MANAGER' AND column\_name='MAINPHONE';

ALTER TABLE MANAGER DROP CONSTRAINT SYS\_C0015948;

ALTER TABLE MANAGER DROP CONSTRAINT SYS\_C0015950;

שינוי שם הטבלה WORKER המקורית ל LIBRARIAN:

*-- Rename the existing worker table to Librarian*

ALTER TABLE Worker RENAME TO Librarian;

עדכון הטיפוס של התכונה mainPhone כך שתתאים ליותר מ 10 תווים (יחזור בכמה מקומות):

ALTER TABLE person MODIFY (mainphone VARCHAR2(15));

טיפול בטבלה LIBRARIAN:

*--change the type of the phone number column of librarian*

ALTER TABLE Librarian RENAME COLUMN workerphone TO workerphoneTemp;

ALTER TABLE Librarian ADD (workerphone VARCHAR2(15));

UPDATE Librarian SET workerphone = TO\_CHAR(workerphoneTemp);

ALTER TABLE Librarian DROP COLUMN workerphoneTemp;

טיפול בטבלה CUSTOMER:

*--change the type of the phone number column of customer*

ALTER TABLE Customer RENAME COLUMN CustomerPhone TO CustomerPhoneTemp;

ALTER TABLE Customer ADD (CustomerPhone VARCHAR2(15));

UPDATE Customer SET CustomerPhone = TO\_CHAR(CustomerPhoneTemp);

ALTER TABLE Customer DROP COLUMN CustomerPhoneTemp;

*-- Create the new Worker table* יצירת הטבלה WORKER

CREATE TABLE Worker

(

MainPhone VARCHAR2(15) NOT NULL,

HireDate DATE NOT NULL,

PRIMARY KEY (MainPhone),

FOREIGN KEY (MainPhone) REFERENCES Person(MainPhone)

);

*-- Update the Librarian table*

*-- Remove the Position column*

ALTER TABLE Librarian DROP COLUMN Position;

טיפול במפתחות מספרי הפלאפון:

UPDATE Librarian L

SET L.WorkerPhone = (

SELECT TO\_CHAR(ROWNUM + 1000000000)

FROM DUAL

WHERE NOT EXISTS (SELECT 1 FROM Person P WHERE P.MainPhone = TO\_CHAR(ROWNUM + 1000000000))

AND NOT EXISTS (SELECT 1 FROM Librarian L2 WHERE L2.WorkerPhone = TO\_CHAR(ROWNUM + 1000000000))

)

WHERE L.WorkerPhone IS NULL;

DROP SEQUENCE membershipid\_seq;

CREATE SEQUENCE membershipid\_seq START WITH 200000000 INCREMENT BY 1;

*-- Add MainPhone to Membership table*

ALTER TABLE Membership ADD MainPhone VARCHAR2(15);

*-- Insert participants who are subscribers into Membership with generated values*

INSERT INTO Membership (MembershipID, StartDate, EndDate, MembershipType, CustomerID, MainPhone)

SELECT membershipid\_seq.NEXTVAL,

TO\_DATE('2023-01-01', 'YYYY-MM-DD') + TRUNC(DBMS\_RANDOM.VALUE(0, 365)),

TO\_DATE('2024-01-01', 'YYYY-MM-DD') + TRUNC(DBMS\_RANDOM.VALUE(0, 365)),

CASE ROUND(DBMS\_RANDOM.VALUE(1, 3))

WHEN 1 THEN 'month'

WHEN 2 THEN 'year'

WHEN 3 THEN '6-month'

END,

-1 as CustomerID,

MainPhone

FROM Participant

WHERE PType = 'subscription';

*-- Update Membership to reference Person*

DELETE FROM Customer

WHERE ROWID IN (

SELECT ROWID FROM (

SELECT ROWID, ROW\_NUMBER() OVER (PARTITION BY CustomerPhone ORDER BY CustomerID) AS rn

FROM Customer

) WHERE rn > 1

);

commit;

*-- Update MainPhone in Membership*

UPDATE Membership SET MainPhone = (SELECT TO\_CHAR(CustomerPhone) FROM Customer WHERE Customer.CustomerID = Membership.CustomerID);

INSERT INTO Person (MainPhone, Pname, Address, Mail, Age)

SELECT

C.CustomerPhone AS MainPhone,

C.CustomerFname || ' ' || C.CustomerLname AS Pname,

'Default Address' AS Address, *-- Placeholder value for Address*

NVL(C.CustomerEmail, C.CustomerFname || '@gmail.com') AS Mail, *-- Use CustomerEmail or generate default email*

30 AS Age *-- Placeholder value for Age*

FROM Customer C

WHERE NOT EXISTS (

SELECT 1 FROM Person P WHERE P.MainPhone = C.CustomerPhone

);

commit;

*-- Add foreign key constraint only if it does not exist*

DECLARE

cnt INTEGER;

BEGIN

SELECT COUNT(\*) INTO cnt

FROM user\_constraints

WHERE table\_name = 'MEMBERSHIP'

AND constraint\_type = 'R'

AND r\_constraint\_name = 'FK\_Membership\_Person';

IF cnt = 0 THEN

EXECUTE IMMEDIATE 'ALTER TABLE Membership ADD CONSTRAINT FK\_Membership\_Person FOREIGN KEY (MainPhone) REFERENCES Person(MainPhone)';

END IF;

END;

*-- Drop the CustomerID column*

ALTER TABLE Membership DROP COLUMN CustomerID;

*-- Ensure MainPhone in Event\_Participant references Person*

BEGIN

EXECUTE IMMEDIATE 'ALTER TABLE Event\_Participant ADD CONSTRAINT FK\_Event\_Participant\_Person FOREIGN KEY (MainPhone) REFERENCES Person(MainPhone)';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE = -2275 THEN

NULL; *-- Constraint already exists, ignore the error*

ELSE

RAISE;

END IF;

END;

ALTER TABLE Participant DROP COLUMN PType;

ALTER TABLE Participant DROP COLUMN UserName;

ALTER TABLE Participant DROP COLUMN PPassword;

*--treat purchase*

*-- Add a new column LibrarianPhone to PURCHASE table*

ALTER TABLE purchase ADD (LibrarianPhone VARCHAR2(15));

ALTER TABLE Librarian ADD CONSTRAINT uq\_workerPhone UNIQUE (workerPhone);

*-- Delete duplicates in Librarian table if the phone number is already in Person table*

DELETE FROM Librarian L

WHERE EXISTS (SELECT 1 FROM Person P WHERE P.MainPhone = L.WorkerPhone);

select\* from purchase;

UPDATE purchase p

SET LibrarianPhone = (

SELECT l.workerphone

FROM Librarian l

WHERE l.WorkerId = p.WorkerId

);

ALTER TABLE purchase ADD CONSTRAINT fk\_purchase\_librarian FOREIGN KEY (LibrarianPhone) REFERENCES Librarian(workerPhone);

*-- Insert Librarian data into Person table*

INSERT INTO Person (MainPhone, Pname, Address, Mail, Age)

SELECT

L.WorkerPhone AS MainPhone,

L.WorkerFname || ' ' || L.WorkerLname AS Pname,

'Default Address' AS Address, *-- Placeholder value*

'defaultemail@example.com' AS Mail, *-- Placeholder value*

30 AS Age *-- Placeholder value*

FROM Librarian L

WHERE L.WorkerPhone IS NOT NULL

AND NOT EXISTS (SELECT 1 FROM Person P WHERE P.MainPhone = L.WorkerPhone);

*-- Insert Librarian data into Worker table*

INSERT INTO Worker (MainPhone, HireDate)

SELECT

L.WorkerPhone AS MainPhone,

L.HireDate AS HireDate

FROM Librarian L

WHERE L.WorkerPhone IS NOT NULL;

*-- Insert Manager data into Worker table*

INSERT INTO Worker (MainPhone, HireDate)

SELECT

M.MainPhone AS MainPhone,

ADD\_MONTHS(SYSDATE, -12 \* M.Seniority) AS HireDate

FROM Manager M;

*-- Delete the Seniority column from Manager table*

ALTER TABLE Manager DROP COLUMN Seniority;

ALTER TABLE Librarian ADD CONSTRAINT FK\_Librarian\_Worker FOREIGN KEY (workerPhone) REFERENCES Worker(MainPhone);

*-- Add a new column CustomerPhone to PURCHASE table*

ALTER TABLE purchase ADD (CustomerPhone VARCHAR2(15));

ALTER TABLE customer RENAME COLUMN CustomerPhone TO MainPhone;

*-- Update CustomerPhone with corresponding MainPhone from Customer table*

UPDATE purchase p

SET CustomerPhone = (

SELECT c.MainPhone

FROM Customer c

WHERE c.CustomerID = p.CustomerID

);

*-- Rename CustomerPhone to MainPhone*

ALTER TABLE Customer ADD CONSTRAINT uq\_customerPhone UNIQUE (MainPhone);

*-- Add foreign key constraint on the new MainPhone column referencing the Person table*

ALTER TABLE purchase ADD CONSTRAINT fk\_purchase\_customer FOREIGN KEY (CustomerPhone) REFERENCES Customer(MainPhone);

ALTER TABLE librarian RENAME COLUMN workerPhone TO MainPhone;

*-- Update LibrarianPhone with corresponding MainPhone from Librarian table*

UPDATE purchase p

SET LibrarianPhone = (

SELECT l.MainPhone

FROM Librarian l

WHERE l.WorkerId = p.WorkerId

);

הוספת אילוצים של מפתחות זרים:

*-- Add foreign key constraint to Manager table*

ALTER TABLE Manager ADD CONSTRAINT fk\_manager\_worker FOREIGN KEY (MainPhone) REFERENCES Worker(MainPhone);

ALTER TABLE Customer ADD CONSTRAINT fk\_customer\_person FOREIGN KEY (MainPhone) REFERENCES Person(MainPhone);

מחיקת עמודות לא נדרשות:

*-- Drop all columns except MainPhone from the CUSTOMER table*

ALTER TABLE Customer DROP COLUMN CustomerFname;

ALTER TABLE Customer DROP COLUMN CustomerLname;

ALTER TABLE Customer DROP COLUMN CustomerEmail;

ALTER TABLE purchase DROP COLUMN WorkerId;

ALTER TABLE purchase DROP COLUMN CustomerID;

ALTER TABLE Customer DROP COLUMN CustomerID;

ALTER TABLE Librarian DROP COLUMN HireDate;

ALTER TABLE Librarian DROP COLUMN WorkerLname;

ALTER TABLE Librarian DROP COLUMN WorkerFname;

ALTER TABLE Librarian DROP COLUMN WorkerId;

**מבטים**

* **על האגף שלנו:**

מבט על האולמות שמכיל פרטים על אולמות האירועים בספריה-

*--create view*

CREATE OR REPLACE VIEW Hall\_View AS

SELECT

HID,

HName,

Place,

Capacity

FROM Hall;

הרצה:



שליפת הנתונים (select\*):

תמונה שמכילה טקסט, חשמל, תצוגה, צילום מסך

התיאור נוצר באופן אוטומטי

**שאילתא א'** – מחזירה את הפרטים על אולמות שהתכולה שלהם גבוהה מ 100 איש:

*--query1*

SELECT

HID,

HName,

Place,

Capacity

FROM Hall\_View

WHERE Capacity > 100;

הרצה:

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי

**שאילתא ב'** – מחזירה את הפרטים על כל האולמות שממוקמים בקומה רביעית ומעלה:

*--query2*

SELECT

HID,

HName,

Place,

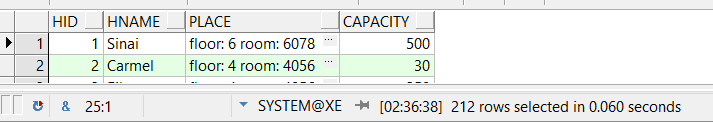
Capacity

FROM Hall\_View

WHERE

TO\_NUMBER(SUBSTR(Place, INSTR(Place,':') + 2, 1))BETWEEN 4 AND 6;

הרצה:



* **על האגף החדש:**

מבט על ישויות הספרים בספריה-

*--create view*

CREATE OR REPLACE VIEW Book\_View AS

SELECT

BookID,

BookName,

Categorys,

Author,

Title,

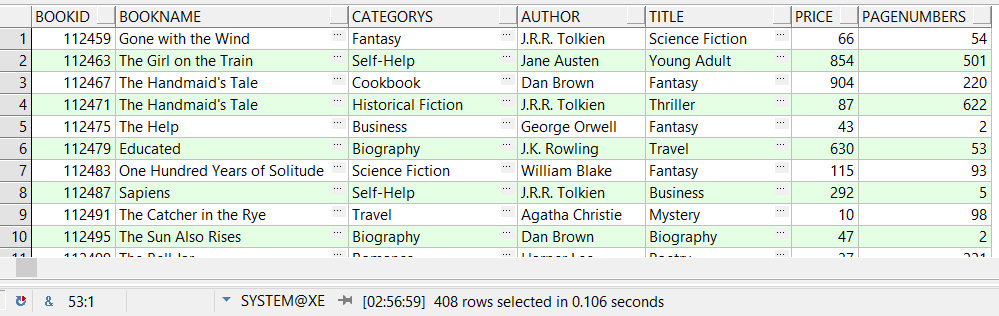
Price,

PageNumbers

FROM Book;

הרצה:  


שליפת הנתונים (select\*):

****

**שאילתא א'** – שאילתא שמחזירה את הספרים שמתחילים באות 'T':

SELECT

BookID,

BookName,

Author,

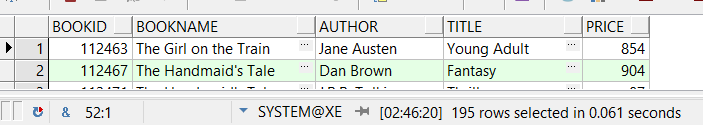
Title,

Price

FROM Book

WHERE BookName LIKE 'T%';

הרצה:



**שאילתא ב'** – מחזירה מידע על הספרים שמחירם בין 10 ל 20 ₪, ומציגה אותם בסדר מחיר יורד.

SELECT

BookID,

BookName,

Author,

Title,

Price

FROM Book\_View

WHERE Price BETWEEN 10 AND 20

ORDER BY Price DESC;

הרצה:

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי