

שלב 1

תיאור הצעת הארגון:

הפרויקט יהיה בארגון בית קולנוע בבסיסי צה"ל

עיסוקו העיקרי של הקולנוע הצבאי הוא קיום הקרנות של סרטים לחיילים. חיילים יכולים לרכוש כרטיסים באמצעות מערכת פנימית ולצפות בסרטים.

בקולנוע ישנם מספר אולמות, כאשר בכל אולם מספר מושבים המחולקים לפי שורות. לכל אולם מספר שורות מסוים. בכל שורה 10 מושבים. השורות ממוספרות מ-1 והלאה והמושבים בכל שורה ממוספרים מ-1 עד 10.

כל מושב מאופיין באופן ייחודי במספר האולם, מספר השורה ומספרו בתוך השורה יחד.

בכל אולם ישנה הקרנה אחת בלבד בכל רגע נתון. ייתכן כי ביום אחד יוקרנו כמה הקרנות באותו אולם וכן יוקרנו כמה הקרנות במקביל באולמות שונים.

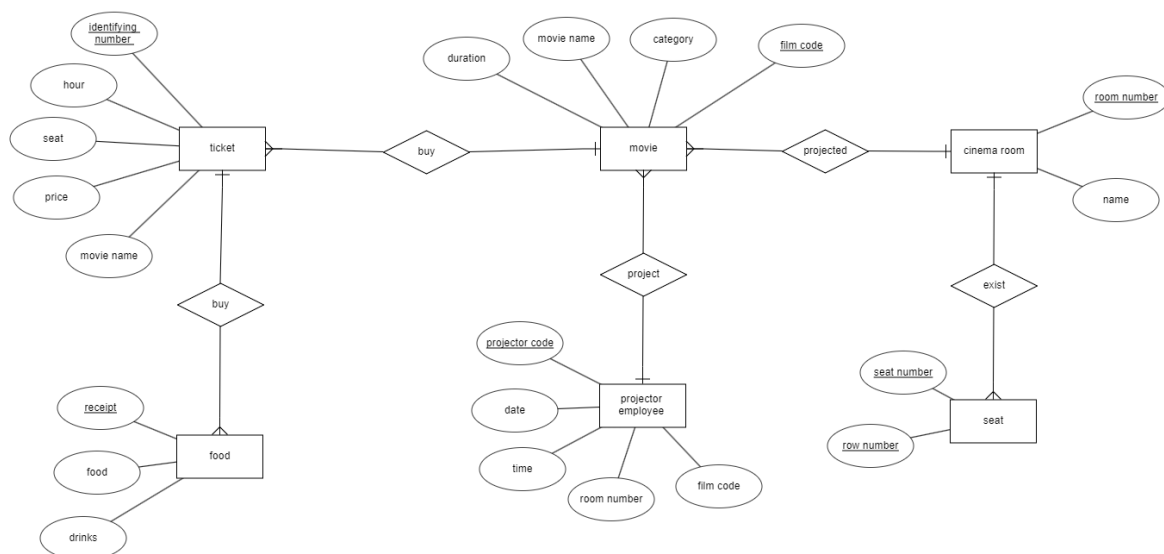
כל סרט יקוטלג בהתאם לתוכנו (דוקומנטרי, לילדים, דרמה, פשע ועוד).

בכל הקרנה מוקרן כמובן סרט אחד בלבד, אך סרט יכול בהחלט להיות מוקרן במספר הקרנות (בלי הגבלה).

כרטיס להקרנה הוא המקשר בין אוכל לבין הקרנה מסוימת ובין מושב. הוא מכיל את פרטי החייל, את המושב בו יישב, את מספר ההקרנה ואת המחיר.

חייבת להיות התאמה בין האולם בו ההקרנה מתקיימת לבין מספר המושב. כמו כן, לא ייתכן שבאותה ההקרנה יהיו שני כרטיסים בעלי אותו מספר מושב.

תרשים ERD



הטבלאות:

אולמות (מספר-אולם, שם-אולם)

מושבים (מספר-שורה, מספר-מושב)

סרטים (קוד-סרט, קטגוריה, שם-סרט, אורך)

עובד הקרנה (מספר-הקרנה, תאריך-הקרנה, שעת-הקרנה, מספר-אולם, קוד הסרט)

כרטיסים (מספר-כרטיס, שעה, מושב, מחיר, שם הסרט)

מזנון(קבלה, אוכל, שתייה)

פירוט הישגיות:

אולם: לכל אולם מספר ייחודי והוא המפתח שלו. ומאפיין נוסף יהיה שם האולם.

מושב: לכל מושב מפתח ייחודי המורכב משלושה פרמטרים: מספר הכיסא מספר השורה, ומספר אולם.

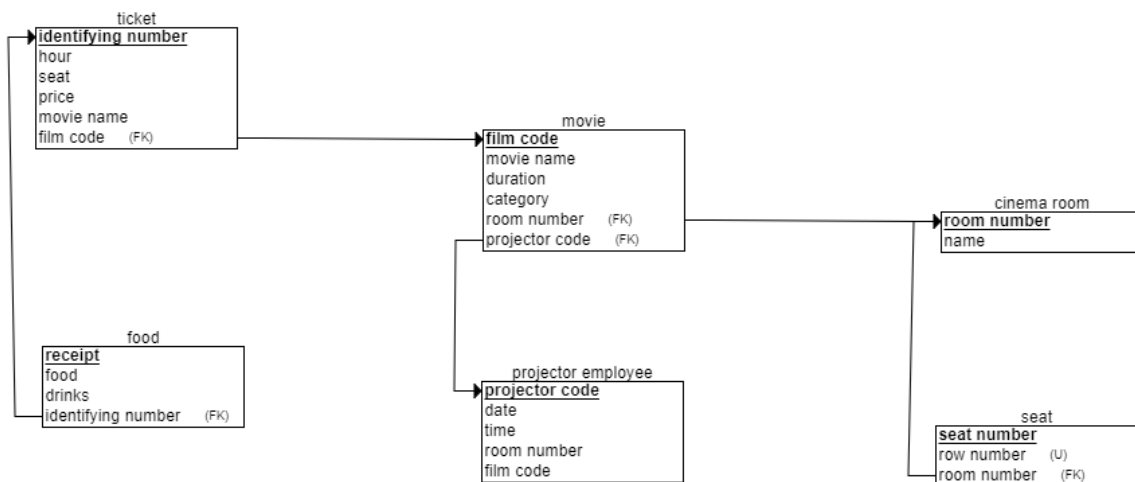
סרט: לכל סרט מספר ייחודי והוא המפתח שלו. ומאפיינים נוספים, קטגוריה, שם הסרט, ואורך הסרט, כך נוכל לחשב כמה זמן ההקרנה וכמה זמן האולם יהיה תפוס.

עובד הקרנה: יש לו מספר יחודי והוא המפתח שלו, תאריך, שעה, מספר חדר, קוד הסרט

כרטיס: מספר יחודי שהוא המפתח שלו, שעה, מיקום(מושב), מחיר, שם הסרט.

מזנון: קבלה ושזה יהיה המפתח, אוכל, ושתייה.

תרשים DSD:



:SQL

```
CREATE TABLE cinema_room
(
    room_number INT NOT NULL,
    name INT NOT NULL,
    PRIMARY KEY (room_number)
);

CREATE TABLE projector_employee
(
    projector_code INT NOT NULL,
    date INT NOT NULL,
    time INT NOT NULL,
    room_number INT NOT NULL,
    film_code INT NOT NULL,
    PRIMARY KEY (projector_code)
);

CREATE TABLE seat
(
    seat_number INT NOT NULL,
    row_number INT NOT NULL,
    room_number INT NOT NULL,
    PRIMARY KEY (seat_number),
    FOREIGN KEY (room_number) REFERENCES cinema_room(room_number),
    UNIQUE (row_number)
);

CREATE TABLE movie
```

```
(
    film_code INT NOT NULL,
    movie_name INT NOT NULL,
    duration INT NOT NULL,
    category INT NOT NULL,
    room_number INT NOT NULL,
    projector_code INT NOT NULL,
    PRIMARY KEY (film_code),
    FOREIGN KEY (room_number) REFERENCES cinema_room(room_number),
    FOREIGN KEY (projector_code) REFERENCES
projector_employee(projector_code)
);

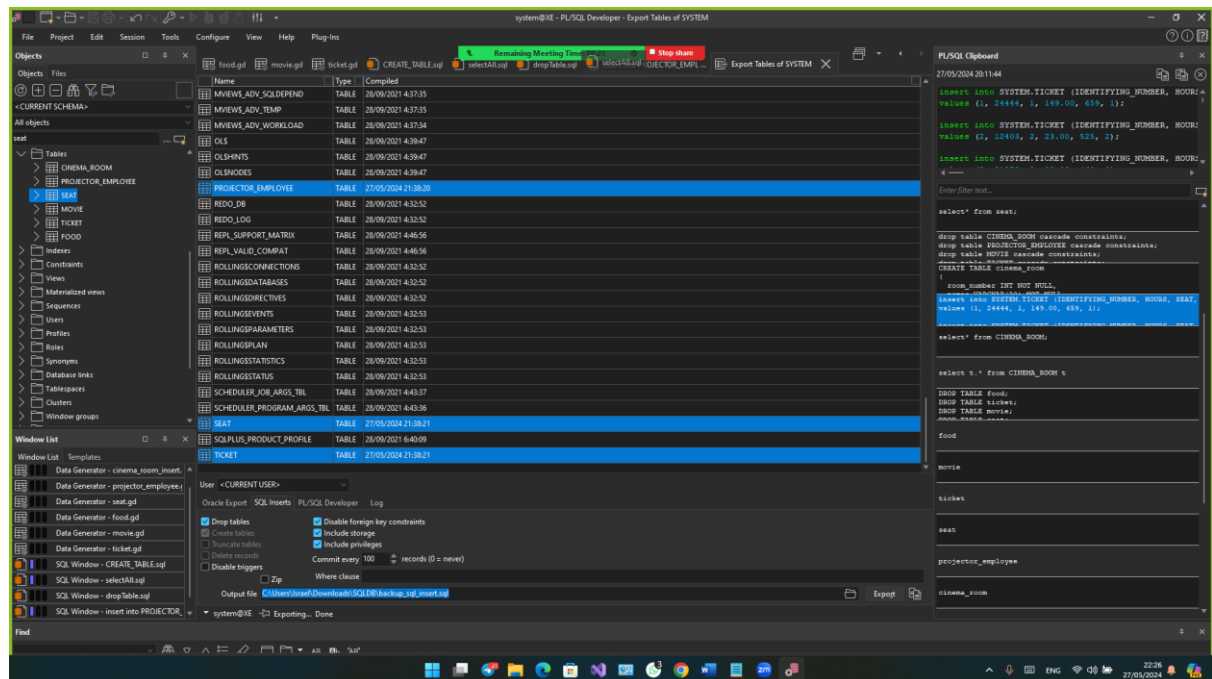
CREATE TABLE ticket
(
    identifying_number INT NOT NULL,
    hour INT NOT NULL,
    seat INT NOT NULL,
    price INT NOT NULL,
    movie_name INT NOT NULL,
    film_code INT NOT NULL,
    PRIMARY KEY (identifying_number),
    FOREIGN KEY (film_code) REFERENCES movie(film_code)
);

CREATE TABLE food
(
    food INT NOT NULL,
    drinks INT NOT NULL,
    receipt INT NOT NULL,
    identifying_number INT NOT NULL,
    PRIMARY KEY (receipt),
    FOREIGN KEY (identifying_number) REFERENCES
ticket(identifying_number)
);
```

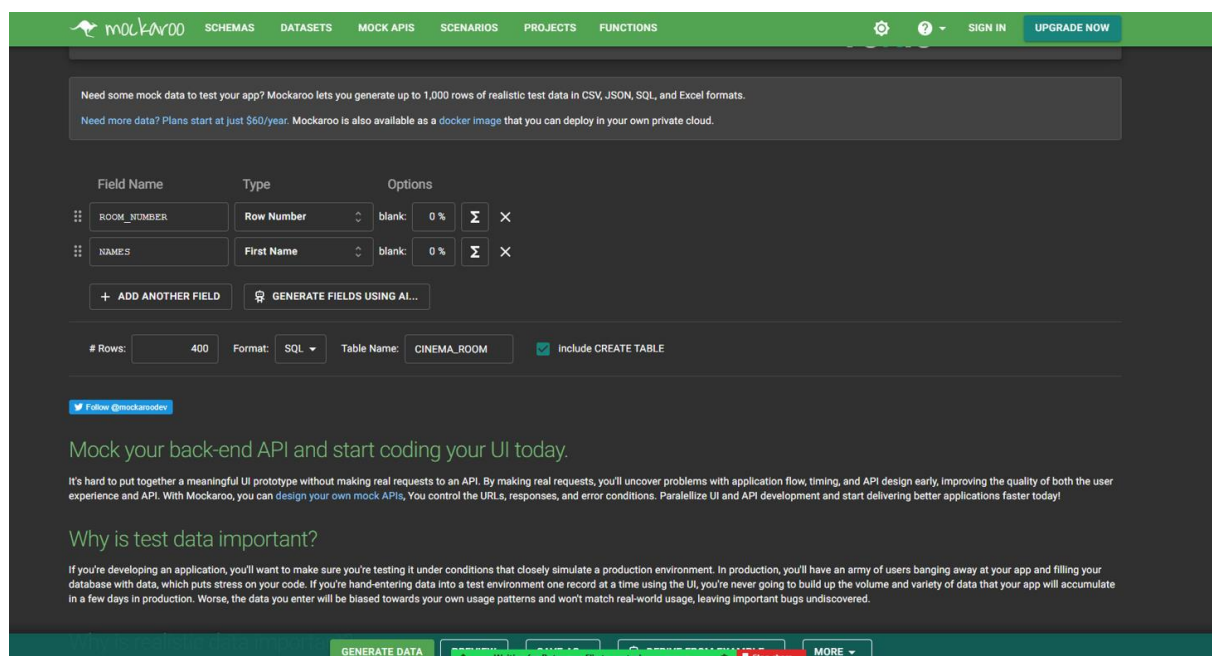
מגישים: ישראל שלמה 315130344
אלי רוטנמר 209572676

מיני פרויקט בבסיסי נתונים

הכנסה פייתון:



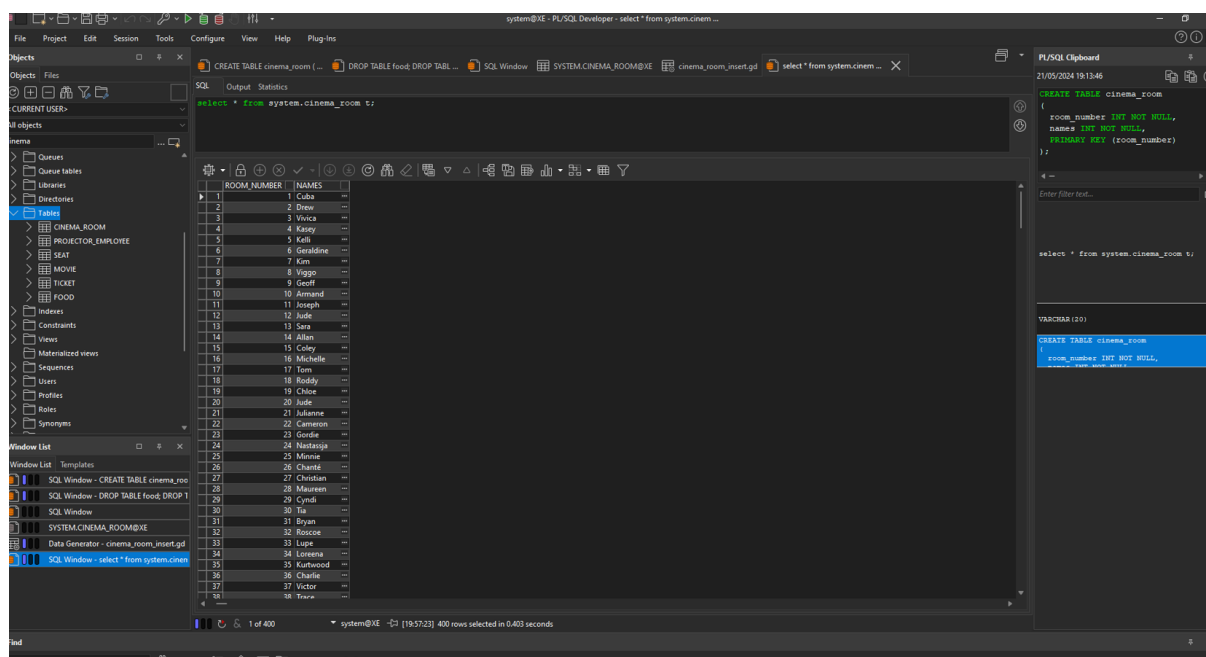
הכנסה mackaroo הקובץ נמצא בגיט:



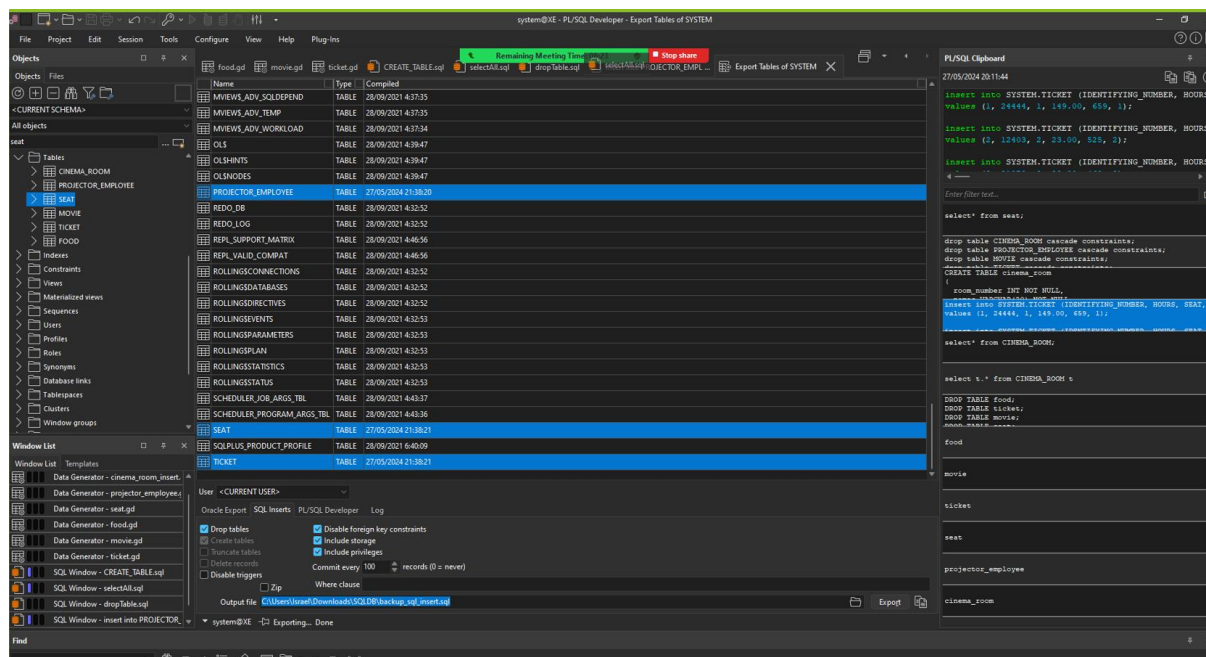
מגישים: ישראל שלמה 315130344
אלי רוטנמר 209572676

מיני פרויקט בבסיסי נתונים

הכנסה ע"י data generator



גיבוי נתונים ושחזור נתונים:

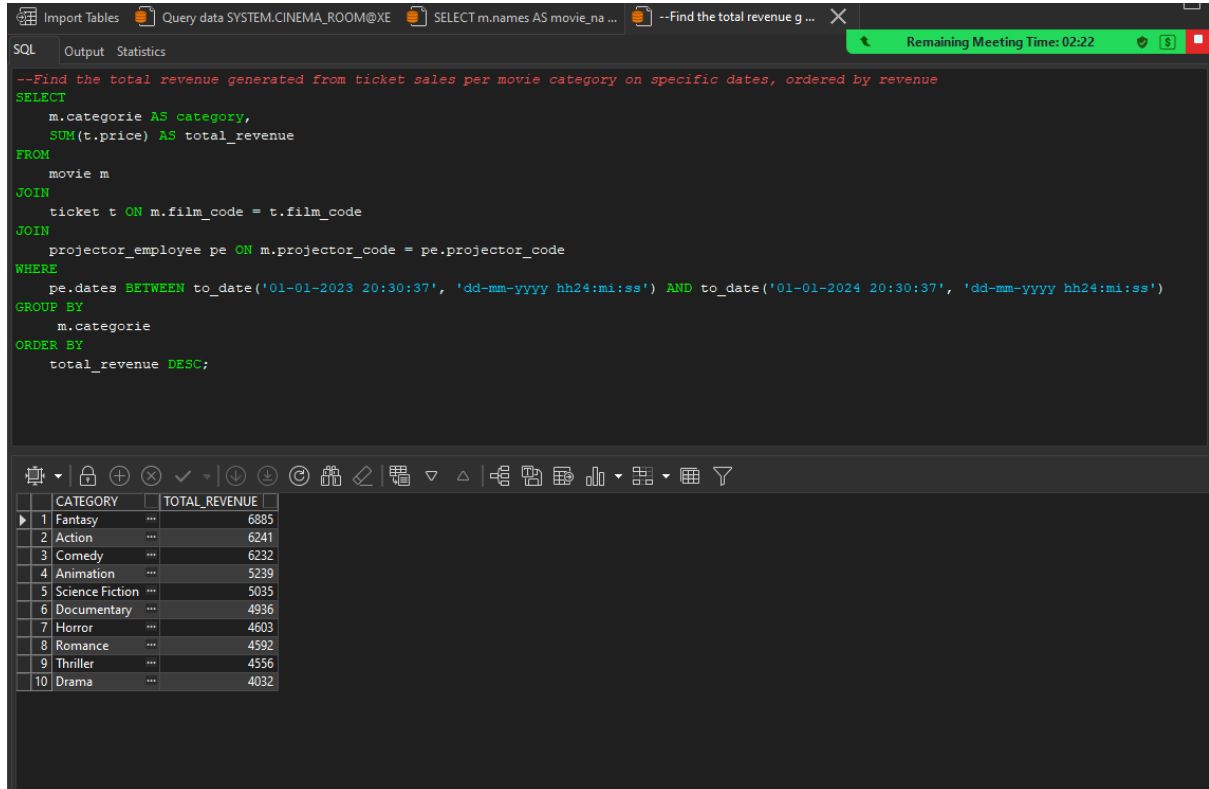


קובץ השחזור נמצא בגיט

שלב ב:

4 שאילתות SELECT

חשב את שיעור התפוסה עבור כל חדר קולנוע וכן כרטיס שנמכר:



The screenshot shows a SQL query editor with a query to find the total revenue generated from ticket sales per movie category on specific dates, ordered by revenue. The query is as follows:

```
--Find the total revenue generated from ticket sales per movie category on specific dates, ordered by revenue
SELECT
  m.categorie AS category,
  SUM(t.price) AS total_revenue
FROM
  movie m
JOIN
  ticket t ON m.film_code = t.film_code
JOIN
  projector_employee pe ON m.projector_code = pe.projector_code
WHERE
  pe.dates BETWEEN to_date('01-01-2023 20:30:37', 'dd-mm-yyyy hh24:mi:ss') AND to_date('01-01-2024 20:30:37', 'dd-mm-yyyy hh24:mi:ss')
GROUP BY
  m.categorie
ORDER BY
  total_revenue DESC;
```

The results table shows the total revenue for each category, ordered by revenue in descending order:

	CATEGORY	TOTAL_REVENUE
1	Fantasy	6885
2	Action	6241
3	Comedy	6232
4	Animation	5239
5	Science Fiction	5035
6	Documentary	4936
7	Horror	4603
8	Romance	4592
9	Thriller	4556
10	Drama	4032

מגישים: ישראל שלמה 315130344
אלי רוטנמר 209572676

מיני פרויקט בבסיסי נתונים

שאלתה שמחזירה את סך ההכנסות שנוצרו ממכירת כרטיסים לכל קטגוריית סרט בתאריכים, לפי הכנסה:

```
--Find the total revenue generated from ticket sales per movie category on specific dates, ordered by revenue
SELECT
  m.categorie AS category,
  SUM(t.price) AS total_revenue
FROM
  movie m
JOIN
  ticket t ON m.film_code = t.film_code
JOIN
  projector_employee pe ON m.projector_code = pe.projector_code
WHERE
  pe.dates BETWEEN to_date('01-01-2023 20:30:37', 'dd-mm-yyyy hh24:mi:ss') AND to_date('01-01-2024 20:30:37', 'dd-mm-yyyy hh24:mi:ss')
GROUP BY
  m.categorie
ORDER BY
  total_revenue DESC;
```

	CATEGORY	TOTAL_REVENUE
1	Fantasy	6885
2	Action	6241
3	Comedy	6232
4	Animation	5239
5	Science Fiction	5035
6	Documentary	4936
7	Horror	4603
8	Romance	4592
9	Thriller	4556
10	Drama	4032

שאלתה שמחזירה את מחיר לפי משך(אורך) הסרט:

```
--receipt price according to movie duration
SELECT
  m.duration AS movie_duration,
  SUM(f.price) AS total_receipt_price
FROM
  food f
JOIN
  ticket t ON f.identifying_number = t.identifying_number
JOIN
  movie m ON t.film_code = m.film_code
GROUP BY
  m.duration
ORDER BY
  m.duration;
```

	MOVIE_DURATION	TOTAL_RECEIPT_PRICE
1	60	93
2	61	80
3	62	254
4	63	206
5	64	29
6	65	168
7	66	122
8	67	104
9	68	11
10	69	61
11	70	316
12	71	244
13	72	108
14	73	151
15	74	50
16	75	187
17	76	188
18	77	54
19	78	81
20	79	124
21	80	91
22	81	64
23	82	9

מגישים: ישראל שלמה 315130344
אלי רוטנמר 209572676

מיני פרויקט בבסיסי נתונים

שאלתה שמחזירה את מספר תחזיות שהתרחשו בין 2023 ל-2024 בכל חדר:

```
--number of projections that happened between 2023 and 2024 in each room
SELECT
  COUNT(pe.projector_code) AS number_of_projections,
  pe.room_number AS room_number
FROM
  projector_employee pe
WHERE
  pe.dates BETWEEN to_date('01-01-2023 20:30:37', 'dd-mm-yyyy hh24:mi:ss') AND to_date('01-01-2024 20:30:37', 'dd-mm-yyyy hh24:mi:ss')
GROUP BY
  pe.room_number
ORDER BY
  number_of_projections DESC;
```

	NUMBER_OF_PROJECTIONS	ROOM_NUMBER
1	28	15
2	27	3
3	27	8
4	25	13
5	24	18
6	22	5
7	22	11
8	22	17
9	21	2
10	20	1
11	19	4
12	19	19
13	18	10
14	18	7
15	17	9
16	16	6
17	14	14
18	14	20
19	14	16
20	12	12

Mute Start Video Participants Chat New share Pause Share

You are screen sharing

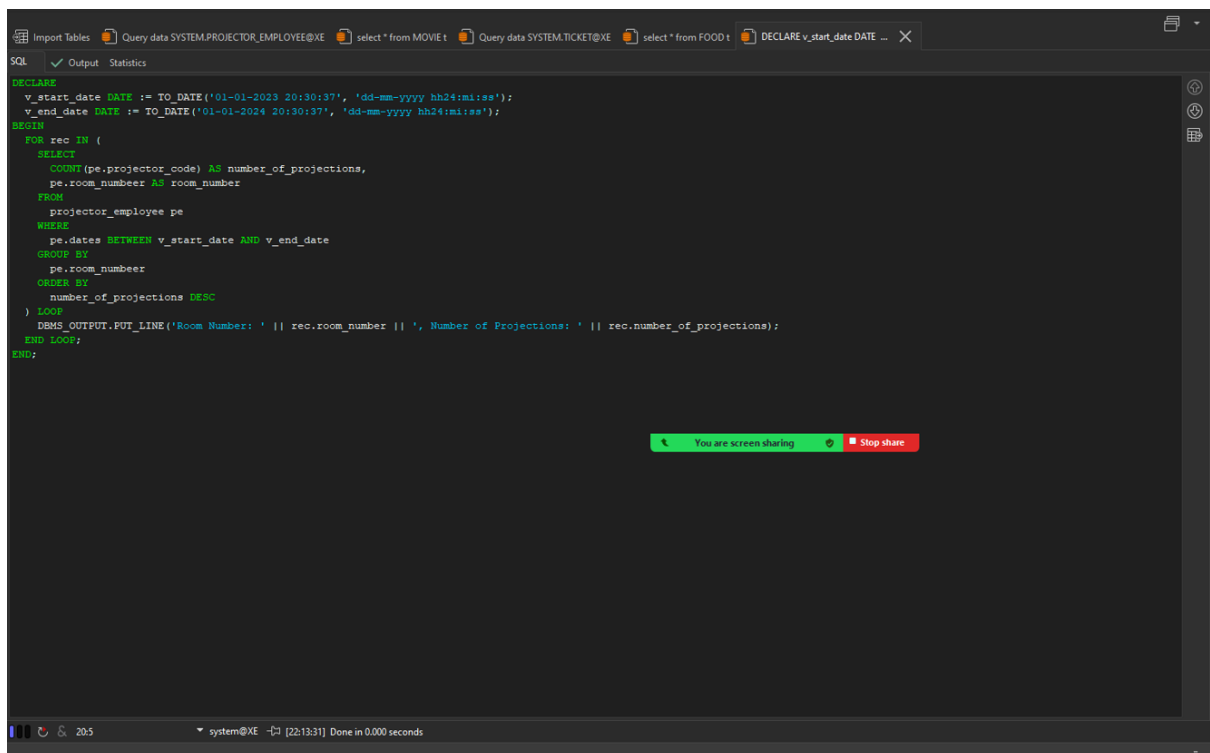
system@XE [19:57:20] 20 rows selected in 0.049 seconds

מגישים: ישראל שלמה 315130344
אלי רוטנמר 209572676

מיני פרויקט בבסיסי נתונים

שאלות עם פרמטרים.

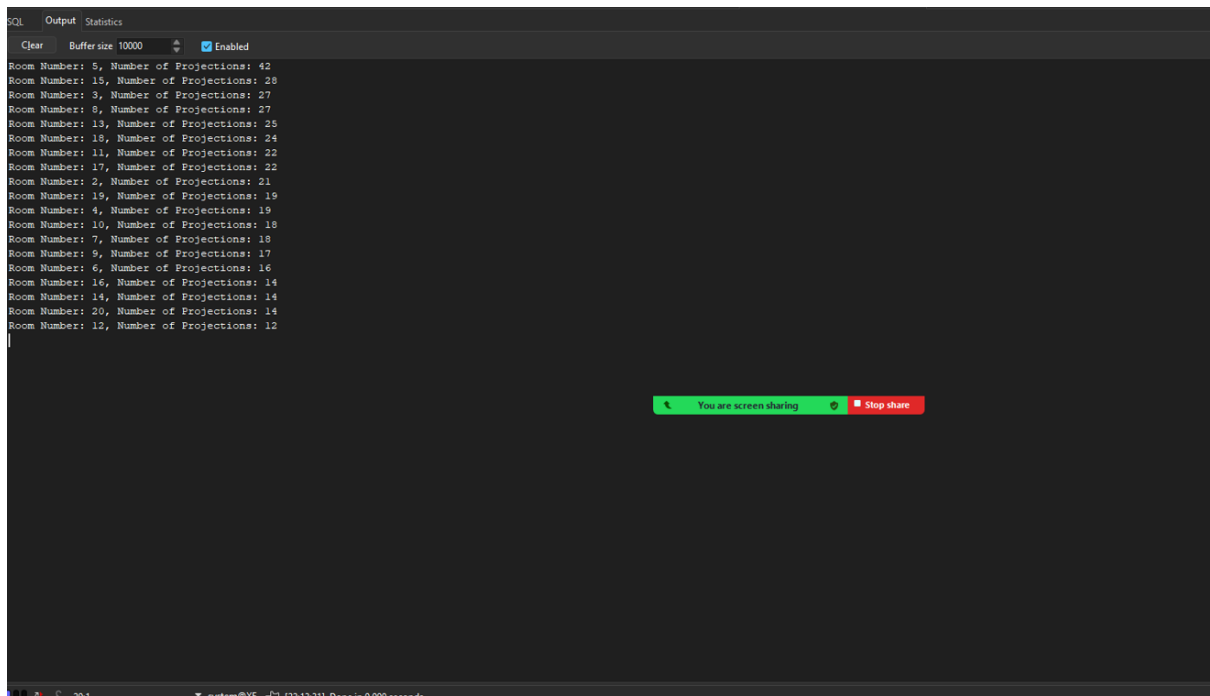
שאלתה 1:



The screenshot shows the SQL Developer interface with a PL/SQL script in the main editor. The script declares two variables, `v_start_date` and `v_end_date`, and uses them in a query. The query counts the number of projections for each room number within a specified date range. The script is as follows:

```
DECLARE
v_start_date DATE := TO_DATE('01-01-2023 20:30:37', 'dd-mm-yyyy hh24:mi:ss');
v_end_date DATE := TO_DATE('01-01-2024 20:30:37', 'dd-mm-yyyy hh24:mi:ss');
BEGIN
FOR rec IN (
SELECT
COUNT(pe.projector_code) AS number_of_projections,
pe.room_number AS room_number
FROM
projector_employee pe
WHERE
pe.dates BETWEEN v_start_date AND v_end_date
GROUP BY
pe.room_number
ORDER BY
number_of_projections DESC
) LOOP
DBMS_OUTPUT.PUT_LINE('Room Number: ' || rec.room_number || ', Number of Projections: ' || rec.number_of_projections);
END LOOP;
END;
```

The status bar at the bottom indicates the script was executed successfully in 0.000 seconds.



The screenshot shows the output of the PL/SQL script in the SQL Developer interface. The output displays the room number and the number of projections for each room, ordered by the number of projections in descending order. The output is as follows:

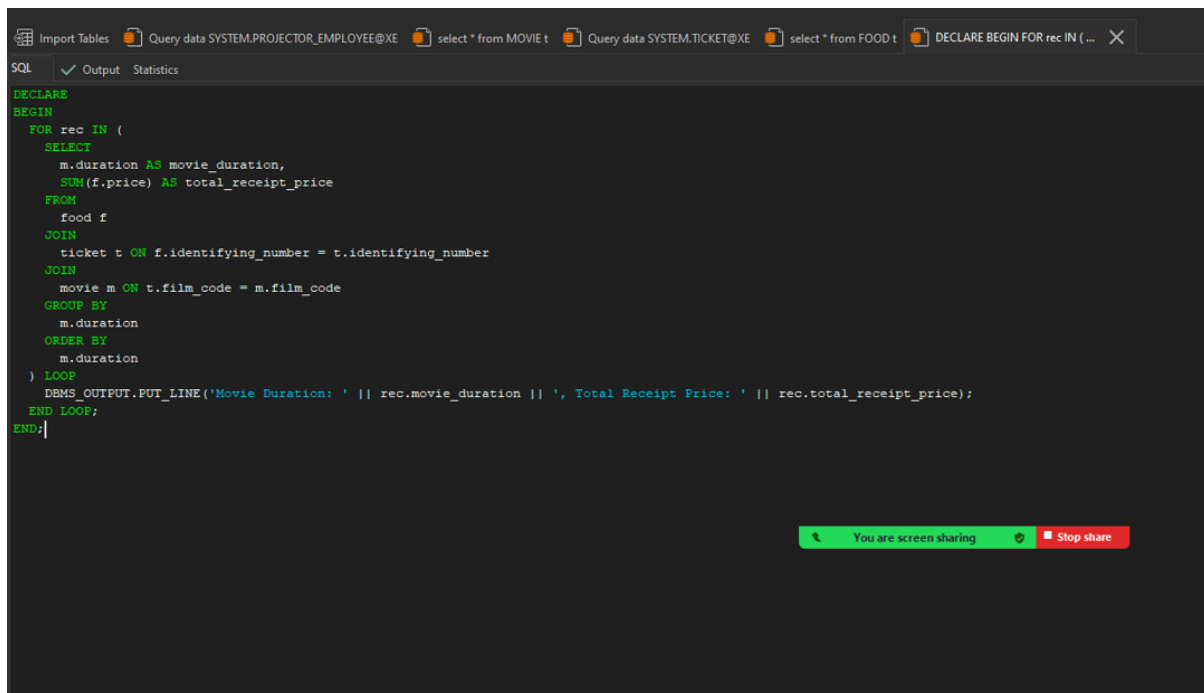
```
Room Number: 5, Number of Projections: 42
Room Number: 15, Number of Projections: 29
Room Number: 3, Number of Projections: 27
Room Number: 8, Number of Projections: 27
Room Number: 13, Number of Projections: 25
Room Number: 18, Number of Projections: 24
Room Number: 11, Number of Projections: 22
Room Number: 17, Number of Projections: 22
Room Number: 2, Number of Projections: 21
Room Number: 19, Number of Projections: 19
Room Number: 4, Number of Projections: 19
Room Number: 10, Number of Projections: 18
Room Number: 7, Number of Projections: 18
Room Number: 9, Number of Projections: 17
Room Number: 6, Number of Projections: 16
Room Number: 16, Number of Projections: 14
Room Number: 14, Number of Projections: 14
Room Number: 20, Number of Projections: 14
Room Number: 12, Number of Projections: 12
```

The status bar at the bottom indicates the script was executed successfully in 0.000 seconds.

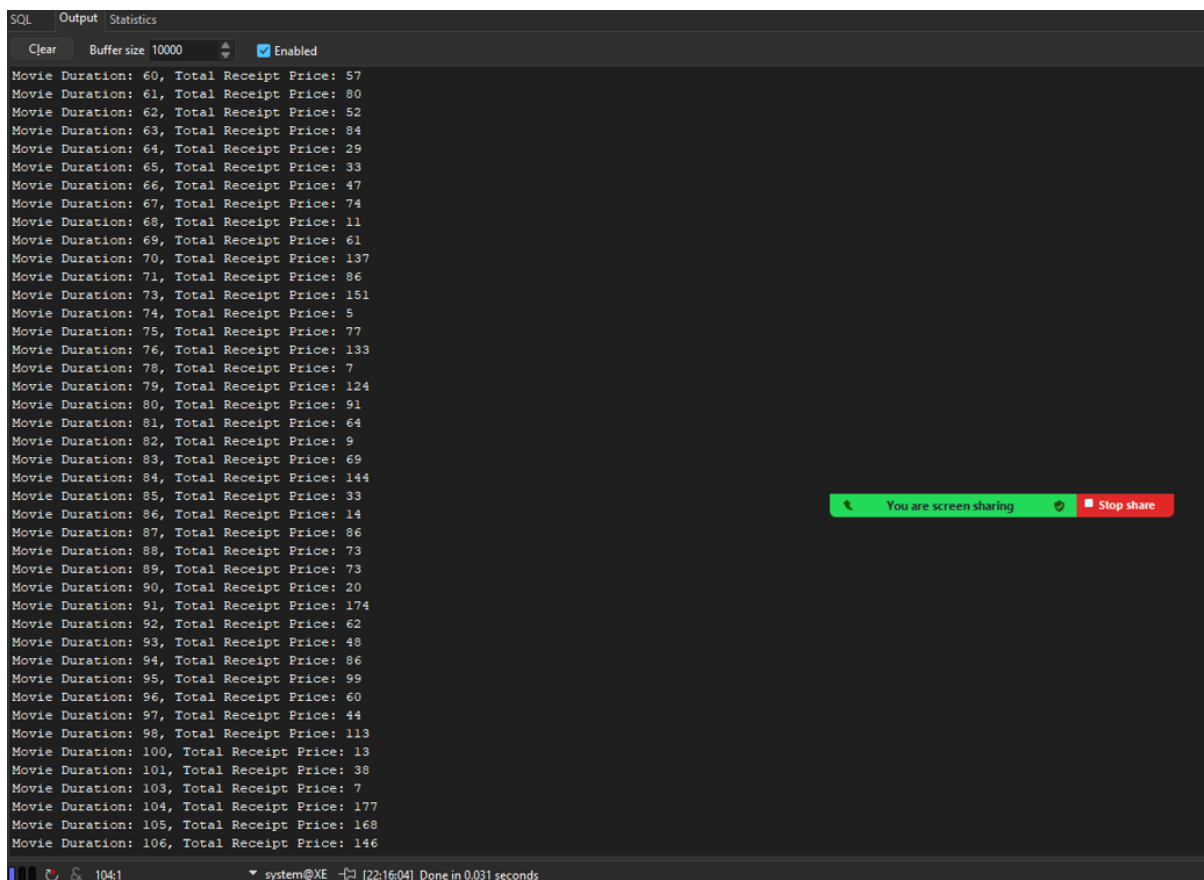
מגישים: ישראל שלמה 315130344
אלי רוטנמר 209572676

מיני פרויקט בבסיסי נתונים

שאלתה 2:



```
DECLARE
BEGIN
  FOR rec IN (
    SELECT
      m.duration AS movie_duration,
      SUM(f.price) AS total_receipt_price
    FROM
      food f
    JOIN
      ticket t ON f.identifying_number = t.identifying_number
    JOIN
      movie m ON t.film_code = m.film_code
    GROUP BY
      m.duration
    ORDER BY
      m.duration
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Movie Duration: ' || rec.movie_duration || ', Total Receipt Price: ' || rec.total_receipt_price);
  END LOOP;
END;
```



```
Movie Duration: 60, Total Receipt Price: 57
Movie Duration: 61, Total Receipt Price: 80
Movie Duration: 62, Total Receipt Price: 52
Movie Duration: 63, Total Receipt Price: 84
Movie Duration: 64, Total Receipt Price: 29
Movie Duration: 65, Total Receipt Price: 33
Movie Duration: 66, Total Receipt Price: 47
Movie Duration: 67, Total Receipt Price: 74
Movie Duration: 68, Total Receipt Price: 11
Movie Duration: 69, Total Receipt Price: 61
Movie Duration: 70, Total Receipt Price: 137
Movie Duration: 71, Total Receipt Price: 86
Movie Duration: 73, Total Receipt Price: 151
Movie Duration: 74, Total Receipt Price: 5
Movie Duration: 75, Total Receipt Price: 77
Movie Duration: 76, Total Receipt Price: 133
Movie Duration: 78, Total Receipt Price: 7
Movie Duration: 79, Total Receipt Price: 124
Movie Duration: 80, Total Receipt Price: 91
Movie Duration: 81, Total Receipt Price: 64
Movie Duration: 82, Total Receipt Price: 9
Movie Duration: 83, Total Receipt Price: 69
Movie Duration: 84, Total Receipt Price: 144
Movie Duration: 85, Total Receipt Price: 33
Movie Duration: 86, Total Receipt Price: 14
Movie Duration: 87, Total Receipt Price: 86
Movie Duration: 88, Total Receipt Price: 73
Movie Duration: 89, Total Receipt Price: 73
Movie Duration: 90, Total Receipt Price: 20
Movie Duration: 91, Total Receipt Price: 174
Movie Duration: 92, Total Receipt Price: 62
Movie Duration: 93, Total Receipt Price: 48
Movie Duration: 94, Total Receipt Price: 86
Movie Duration: 95, Total Receipt Price: 99
Movie Duration: 96, Total Receipt Price: 60
Movie Duration: 97, Total Receipt Price: 44
Movie Duration: 98, Total Receipt Price: 113
Movie Duration: 100, Total Receipt Price: 13
Movie Duration: 101, Total Receipt Price: 38
Movie Duration: 103, Total Receipt Price: 7
Movie Duration: 104, Total Receipt Price: 177
Movie Duration: 105, Total Receipt Price: 168
Movie Duration: 106, Total Receipt Price: 146
```

שאלה 3:

```
DECLARE
v_start_date DATE := TO_DATE('01-01-2023 20:30:37', 'dd-mm-yyyy hh24:mi:ss');
v_end_date DATE := TO_DATE('01-01-2024 20:30:37', 'dd-mm-yyyy hh24:mi:ss');
BEGIN
FOR rec IN (
SELECT
m.categorie AS category,
SUM(t.price) AS total_revenue
FROM
movie m
JOIN
ticket t ON m.film_code = t.film_code
JOIN
projector_employee pe ON m.film_code = pe.projector_code
WHERE
pe.dates BETWEEN v_start_date AND v_end_date
GROUP BY
m.categorie
ORDER BY
total_revenue DESC
) LOOP
DBMS_OUTPUT.PUT_LINE('Category: ' || rec.category || ', Total Revenue: ' || rec.total_revenue);
END LOOP;
END;
```

SQL Output Statistics

Clear Buffer size 10000 ☒ Enabled

Category: Fantasy, Total Revenue: 6885
Category: Action, Total Revenue: 6241
Category: Comedy, Total Revenue: 6232
Category: Animation, Total Revenue: 5239
Category: Science Fiction, Total Revenue: 5035
Category: Documentary, Total Revenue: 4936
Category: Horror, Total Revenue: 4603
Category: Romance, Total Revenue: 4592
Category: Thriller, Total Revenue: 4556
Category: Drama, Total Revenue: 4032

2 שאלות המחיקה:

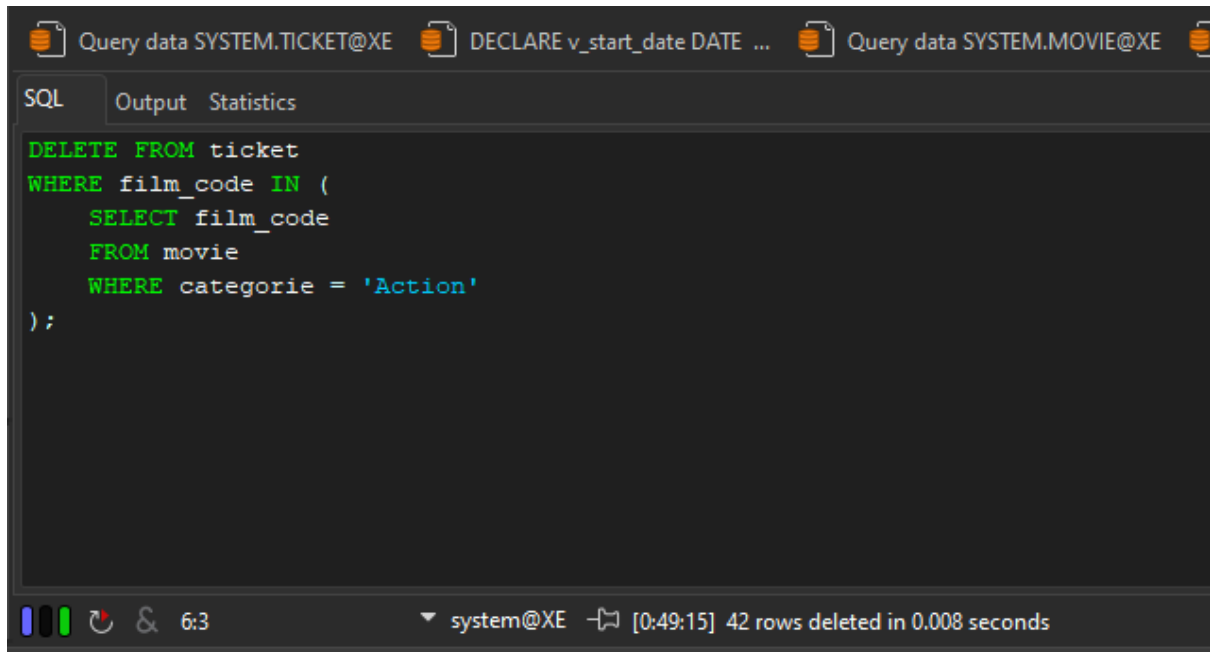
שאלתה 1 מוחקת סוג מסוים של מזון:

```
DELETE FROM food
WHERE food_items LIKE '%Salmon%';|
```

system@XE [21:18:52] 158 rows deleted in 0.039 seconds

שאלתה 2:

מחק את כל הכרטיסים לקטגוריית סרטים ספציפית:



The screenshot shows the Oracle SQL Developer interface. At the top, there are three tabs: 'Query data SYSTEM.TICKET@XE', 'DECLARE v_start_date DATE ...', and 'Query data SYSTEM.MOVIE@XE'. The 'SQL' tab is active, displaying the following SQL query:

```
DELETE FROM ticket
WHERE film_code IN (
  SELECT film_code
  FROM movie
  WHERE categorie = 'Action'
);
```

At the bottom of the window, the status bar indicates the user is 'system@XE', the session duration is '[0:49:15]', and the execution result is '42 rows deleted in 0.008 seconds'.

2 שאלות העדכון:

שאלתה של עדכון והחלפה סוג מסוים של מזון:

```
UPDATE food
SET food_items = REPLACE(food_items, 'Soda', 'Coke')
WHERE food_items LIKE '%Soda%';
```

שאלתה שמעדכנת את מספר חדרי הקולנוע:

```
UPDATE projector_employee  
SET room_numbeer = 5  
WHERE room_numbeer = 1;
```

Waiting for R

3:24

system@XE [21:32:12] 20 rows updated in 0.006 seconds

