
מיני פרויקט בבסיסי נתונים

נושא: פנאי - קולנוע



מגישות:

וינר רחלי – 325265221, viderracheli@gmail.com
מייפן נעמה – 214429193, naamameypen@gmail.com

תוכן עניינים

3	תיאור מילולי של המערכת.....
3	תיאור מילולי של הטבלאות.....
4	הפונקציונאליות העיקרית של המערכת
5	תרשים ERD.....
6	תרשימי DSD.....
7	הצגת Create tables ופקודות Desc.....
10	הכנסת נתונים - Data generator.....
11	הכנסת נתונים - Python Script.....
12	הכנסת נתונים - mockaroo.....
13	גיבוי ושיחזור
15	שאילתות Select ללא פרמטרים.....
18	שאילתות Delet.....
21	שאילתות Update.....
24	שאילתות Select עם פרמטרים
29	אילוצים
34	פונקציות
40	פרוצדורות
47	תוכניות

דוח הפרוייקט שלב א

פרוייקט על ארגון בתחום הפנאי פנאי - קולנוע.

תיאור המערכת:

מערכת ניהול הקולנוע נועדה לטפל בהיבטים התפעוליים של בית קולנוע. זה מקל על ניהול סרטים, זמני הצגה, קולנוע, מנויים ומכירת כרטיסים. המערכת שמה לה למטרה לייעל את תפעול הקולנוע, לשפר את חווית הלקוח ולייעל את ניצול המשאבים. על ידי שמירה על רישומים מפורטים ומאפשרת תזמון יעיל, המערכת מבטיחה תפקוד חלק ומשפרת את היעילות הכוללת של פעולות הקולנוע.

נתונים - טבלאות:

1. **סרטים:** מזהה סרט, כותרת, ז'אנר, במאי, תאריך יציאה, משך, דירוג.

טבלה זו מאחסנת מידע על הסרטים המוצגים בקולנוע, כולל תעודת הזהות, הכותרת, הז'אנר, הבמאי, תאריך היציאה, משך הזמן והדירוג שלהם.

2. **תאריכים:** DateID, HourH, DayD, DateD.

טבלה זו עוקבת אחר התאריכים והשעות שבהם הסרטים מתוכננים להיות מוצגים.

3. **אולמות:** TheaterID, TheaterName, Capacity.

טבלה זו מכילה מידע על בתי הקולנוע בתוך הקולנוע, כולל תעודת הזהות, השם ויכולת הישיבה של כל אולם.

4. **לוחות זמנים:** ScheduleID, TheaterID, AvailableSeats, MovieID, DateID.

טבלה זו מקשרת בין סרטים, תאריכים ובתי קולנוע, ומציינת איזה סרט מוצג באיזה קולנוע באיזו שעה וכמה מושבים זמינים לכל הצגה.

5. **מנויים:** מזהה מנוי, אחוזי הנחה, תאריך תפוגה.

טבלה זו מאחסנת מידע על המנויים, כולל תעודת הזהות שלהם, אחוז ההנחה ותאריך תפוגה של המנוי.

6. **מכירות כרטיסים:** TicketID, Price, IsSold, ScheduleID, SubscriberID.

טבלה זו מתעדת את מכירת הכרטיסים, כולל מזהה כל כרטיס, מחיר, סטטוס מכירה והקשרים עם לוחות זמנים ומנויים ספציפיים.

פונקציות עיקריות של המערכת:

1. ניהול סרטים:

- הוספה, עדכון והסרה של סרטים.
- מעקב אחר פרטי סרט כגון ז'אנר, במאי, תאריך יציאה, משך ודירוג.

2. תזמון מועדי הופעה:

- הגדרת זמני הקרנה לסרטים.
- הקצאת בתי קולנוע לסרטים שונים.
- ניהול המושבים הפנויים לכל שעת הופעה.

3. ניהול אולמות:

- שמירה על מידע על כל אולם בתוך הקולנוע.
- ניהול קיבולת וזמינות האולם.

4. ניהול מנויים:

- ניהול מידע מנויים.
- החלת הנחות למנויים.
- מעקב אחר תאריכי תפוגה של מנוי.

5. מכירת כרטיסים:

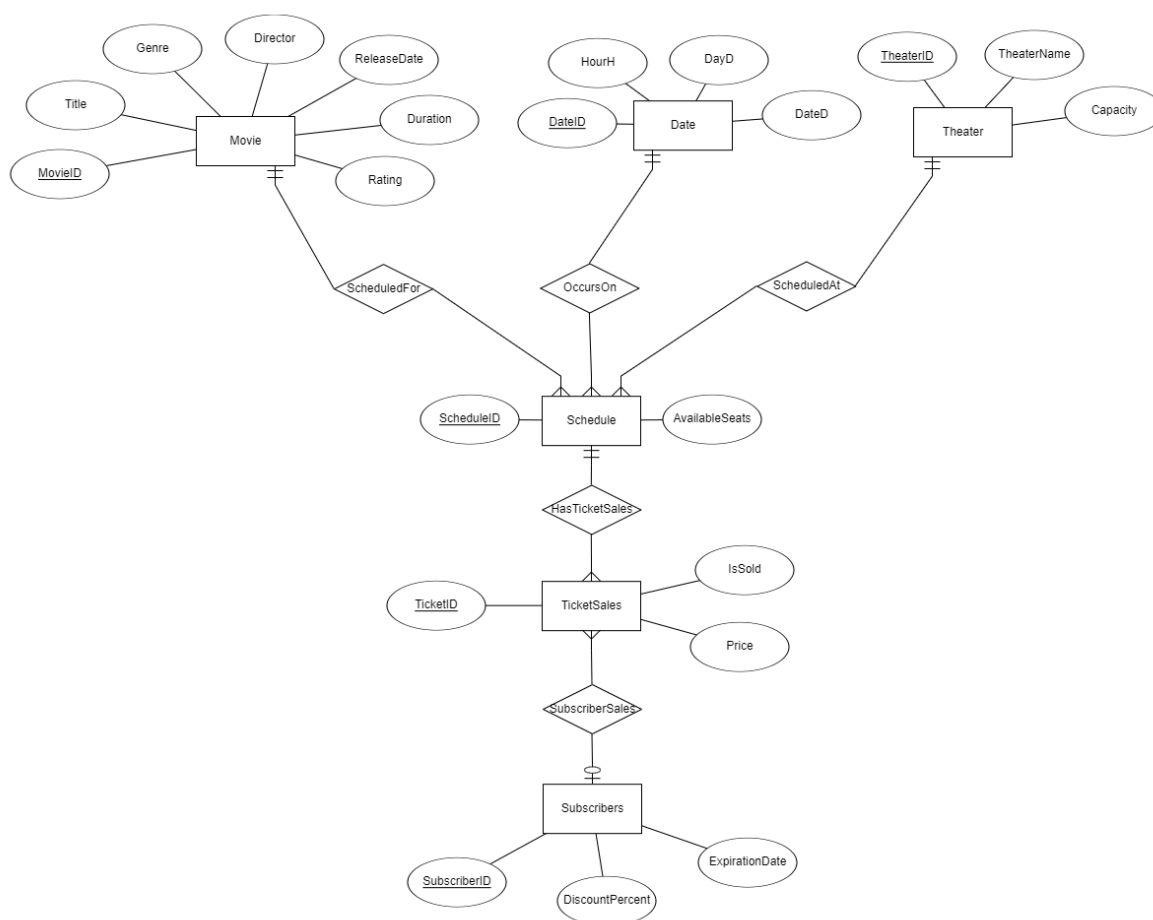
- מכירת כרטיסים למועדי הופעה קבועים.
- מעקב אחר מכירת כרטיסים וזמינות.
- החלת הנחות מנויים על מכירת כרטיסים.

6. ניהול תאריך וזמן:

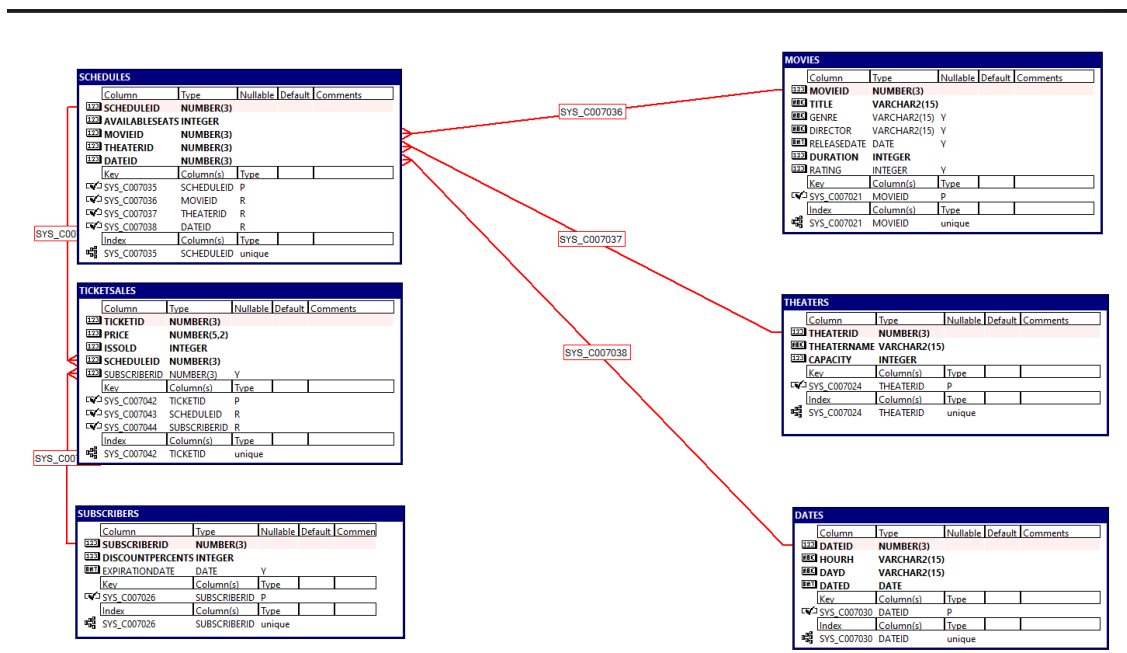
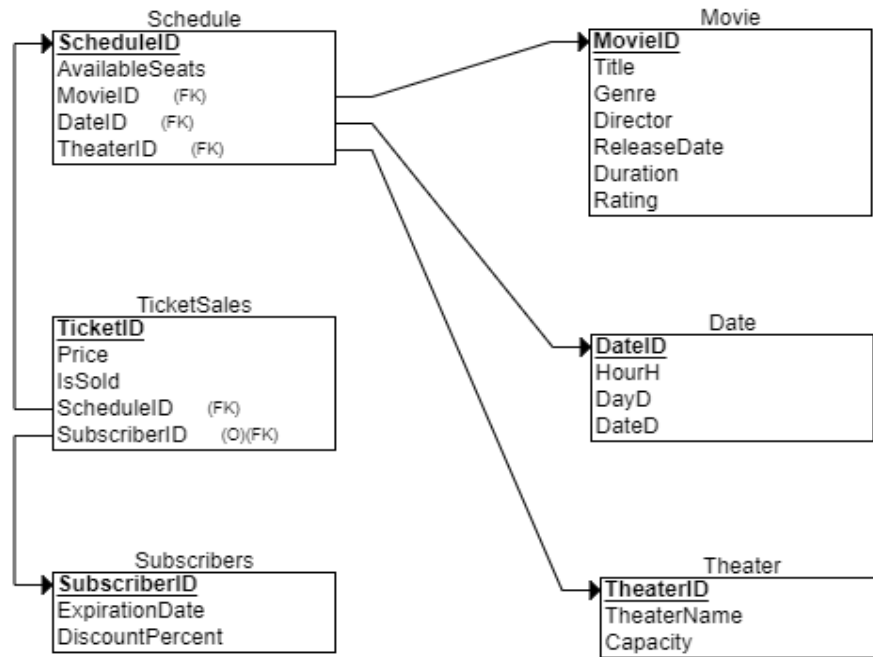
- רישום של כל התאריכים והשעות שבהם מוצגים סרטים.
- הבטחת תזמון מדויק והימנעות מעימותים.

תרשימי ה ERD ו ה DSD:

ERD:



DSD:



הצגת Create tables ופקודות Desc:

1. :Movies

```
CREATE TABLE Movies (  
    MovieID NUMERIC(3) PRIMARY KEY,  
    Title VARCHAR2(100) NOT NULL,  
    Genre VARCHAR2(100),  
    Director VARCHAR2(100),  
    ReleaseDate DATE,  
    Duration INT NOT NULL,  
    Rating INT  
);
```

```
SQL> desc Movies;  
Name          Type          Nullable Default Comments  
-----  
MOVIEID       NUMBER(3)  
TITLE         VARCHAR2(100)  
GENRE         VARCHAR2(100) Y  
DIRECTOR      VARCHAR2(100) Y  
RELEASEDATE   DATE          Y  
DURATION      INTEGER  
RATING        INTEGER      Y
```

2. :Dates

```
CREATE TABLE Dates (  
    DateID NUMERIC(3) PRIMARY KEY,  
    HourH VARCHAR2(100) NOT NULL,  
    DayD VARCHAR2(100) NOT NULL,  
    DateD DATE NOT NULL  
);
```

```
SQL> desc Schedules  
Name          Type          Nullable Default Comments  
-----  
SCHEDULEID    NUMBER(3)  
THEATERID     NUMBER(3)  
AVAILABLESEATS INTEGER  
MOVIEID       NUMBER(3)  
DATEID        NUMBER(3)
```

:Theaters .3

```
CREATE TABLE Theaters (  
    TheaterID NUMERIC(3) PRIMARY KEY,  
    TheaterName VARCHAR2(30) NOT NULL,  
    Capacity INT NOT NULL  
);
```

```
SQL> desc theaters  
Name          Type          Nullable Default Comments  
-----  
THEATERID     NUMBER(3)  
THEATERNAME   VARCHAR2(100)  
CAPACITY      INTEGER
```

:Schedules .4

```
CREATE TABLE Schedules (  
    ScheduleID NUMERIC(3) PRIMARY KEY,  
    AvailableSeats INT NOT NULL,  
    MovieID NUMERIC(3) NOT NULL,  
    TheaterID NUMERIC(3) NOT NULL,  
    DateID NUMERIC(3) NOT NULL,  
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID),  
    FOREIGN KEY (TheaterID) REFERENCES Theaters(TheaterID),  
    FOREIGN KEY (DateID) REFERENCES Dates(DateID)  
);
```

```
SQL> desc Schedules  
Name          Type          Nullable Default Comments  
-----  
SCHEDULEID    NUMBER(3)  
THEATERID     NUMBER(3)  
AVAILABLESEATS INTEGER  
MOVIEID       NUMBER(3)  
DATEID        NUMBER(3)
```

:Subscribers .5


```
CREATE TABLE Subscribers (
  SubscriberID NUMERIC(3) PRIMARY KEY,
  DiscountPercents INT NOT NULL,
  ExpirationDate DATE
);
```

```
SQL> desc Subscribers
Name                Type                Nullable Default Comments
-----
SUBSCRIBERID        NUMBER(3)
DISCOUNTPERCENTS  INTEGER
EXPIRATIONDATE      DATE                Y
```

:TicketSales .6

```
CREATE TABLE TicketSales (
  TicketID NUMERIC(3) PRIMARY KEY,
  Price NUMERIC(5, 2) NOT NULL,
  IsSold CHAR(1) DEFAULT 'N' CHECK (IsSold IN ('Y', 'N')) NOT NULL,
  ScheduleID NUMERIC(3) NOT NULL,
  SubscriberID NUMERIC(3),
  FOREIGN KEY (ScheduleID) REFERENCES Schedules(ScheduleID),
  FOREIGN KEY (SubscriberID) REFERENCES Subscribers(SubscriberID)
);
```

```
SQL> desc TicketSales
Name                Type                Nullable Default Comments
-----
TICKETID            NUMBER(3)
PRICE               NUMBER(5,2)
ISSOLD              CHAR(1)                'N'
SCHEDULEID          NUMBER(3)
SUBSCRIBERID        NUMBER(3)            Y
```

הכנסת נתונים לטבלאות ב3 שיטות:

1. Data generator: (TicketSales, Schedules)

TICKETSALES					
Owner		Table		Number of records	
RACHELI		TICKETSALES		400	
Name	Type	Size		Data	Master
TICKETID	NUMBER	3		Sequence(100)	...
PRICE	NUMBER	5,2		List(30, 30, 30, 30, 50, 90)	...
ISSOLD	CHAR	1		List(N)	...
SCHEDULEID	NUMBER	3		List(select scheduleid from schedules)	...
SUBSCRIBERID	NUMBER	3		List(select subscriberid from subscribers)	...
*					...

SCHEDULES					
Owner		Table		Number of records	
RACHELI		SCHEDULES		400	
Name	Type	Size		Data	
SCHEDULEID	NUMBER	3		Sequence(100)	
AVAILABLESEATS	NUMBER			List(0)	
MOVIEID	NUMBER	3		List(select MOVIEID from MOVIES)	
THEATERID	NUMBER	3		List(select THEATERID from THEATERS)	
DATEID	NUMBER	3		List(select DATEID from DATES)	
*					

```

1195 insert into RACHELI.SCHEDULES (SCHEDULEID, AVAILABLESEATS, MOVIEID, THEATERID, DATEID)
1196 values (498, 0, 424, 297, 190);
1197
1198 insert into RACHELI.SCHEDULES (SCHEDULEID, AVAILABLESEATS, MOVIEID, THEATERID, DATEID)
1199 values (499, 0, 120, 428, 228);
1200
1201 UPDATE Schedules s
1202 SET s.AvailableSeats = (
1203     SELECT t.Capacity
1204     FROM Theaters t
1205     WHERE t.TheaterID = s.TheaterID
1206 );
1207
1208 commit;
1209

```

לאחר הכנסת מקומות הישיבה כ 0 בכל הטבלאות, הוספנו בסוף הקובץ עדכון לערך תכולה, שיתאים לאולם שנבחר על פי ה TheaterID ניקח את התכולה של כל שורה בטבלה Theaters.

2. Python Script: (Theaters, Dates)

```
1 def read_flow_names(filename):
2     with open(filename, 'r') as file:
3         flowers = [line.strip() for line in file.readlines()]
4     return flowers
5
6 def generate_plsql_script(flowers, num_records, output_filename):
7     theater_id_start = 111
8     min_capacity = 20
9     max_capacity = 300
10
11     with open(output_filename, 'w') as file:
12         file.write("BEGIN\n")
13         for i in range(num_records):
14             theater_id = theater_id_start + i
15             theater_name = flowers[i % len(flowers)]
16             capacity = random.randint(min_capacity, max_capacity)
17             file.write(f"INSERT INTO THEATERS (THEATERID, THEATERNAME, CAPACITY) VALUES ({theater_id}, '{theater_name}', {capacity});\n")
18         file.write("END;\n")
19     file.write("\n")
20
21 if __name__ == "__main__":
22     flowers = read_flow_names('flower_names.txt')
23     num_records = 400
24     output_filename = 'insert_theaters.sql'
25     generate_plsql_script(flowers, num_records, output_filename)
26     print(f"PL/SQL script '{output_filename}' generated successfully.")
```

```
1 from datetime import datetime, timedelta
2
3 # Function to generate sequential dates
4 def generate_sequential_dates(start_date, num_days):
5     return [start_date + timedelta(days=i) for i in range(num_days)]
6
7 # Define hours and days
8 hours = ["14:00", "17:00", "20:00", "23:00"]
9
10 # Start date for the records
11 start_date = datetime(2024, 6, 1)
12 num_days = 120 # Number of days to generate
13 total_records = 400 # Total number of unique records to generate
14
15 # Generate sequential dates
16 sequential_dates = generate_sequential_dates(start_date, num_days)
17
18 # Calculate how many records per day are needed
19 records_per_day = total_records // num_days
20
21 # Generate 400 unique records
22 records = []
23 ur = set()
24 date_id = 100 # Starting DateID
25
26 for day_date in sequential_dates:
27     day = day_date.strftime('%a').upper()
28     date_d = day_date.strftime('%d-%m-%Y')
29     hour_count = 0
30
```

```

41 for hour in hours:
42     if len(records) >= total_records:
43         break
44     # Use a tuple to ensure uniqueness
45     record = (date_id, hour, day, date_d)
46     unique_key = (hour, day, date_d)
47
48     # Add record to the set if the unique combination does not exist
49     if unique_key not in ur:
50         ur.add(unique_key)
51         records.append(record)
52         date_id += 1
53         hour_count += 1
54
55 # Write the insert statements to a SQL file
56 with open('Date_insertion.sql', 'w') as file:
57     for record in records:
58         date_id, hour, day, date_d = record
59         insert_statement = f"INSERT INTO DATES (DATEID, HOURH, DAYD, DATED) VALUES ({date_id}, '{hour}', '{day}', _to_date('{date_d}', 'DD-MM-YYYY'));"
60         file.write(insert_statement + '\n')
61
62

```

3. mockaroo: (Subscribers, Movies)

Subscribers

Field Name	Type	Options
SubscriberIDCKETID	Digit Sequence	### blank: 0% Σ ×
DiscountPercents	Custom List	10,20,30,40,50 random blank: 0% Σ ×
ExpirationDate	Datetime	05/20/2024 to 05/20/2030 format: dd-mm-yyyy blank: 0% Σ ×

+ ADD ANOTHER FIELD GENERATE FIELDS USING AI...

Rows: 400 Format: SQL Table Name: Subscribers ☐ Include CREATE TABLE

Append Dataset: choose a dataset...

Movies

Field Name	Type	Options
MovieID	Digit Sequence	### blank: 0% Σ ×
Title	Movie Title	blank: 0% Σ ×
Genre	Movie Genres	blank: 0% Σ ×
Director	Full Name	blank: 0% Σ ×
ReleaseDate	Datetime	05/20/2000 to 05/20/2024 format: m/d/yyyy blank: 0% Σ ×
Duration	Number	min: 60 max: 180 decimals: 0 blank: 0% Σ ×
Rating	Number	min: 1 max: 10 decimals: 0 blank: 0% Σ ×

+ ADD ANOTHER FIELD GENERATE FIELDS USING AI...

Rows: 100 Format: SQL Table Name: Movies ☐ Include CREATE TABLE

עבור שדה מטיפוס תאריך זה הקוד שכתבנו:

```
code(concat(code("to_date('",this, code("'", 'dd-mm-yyyy')"))
```

הערה:

הערה:

הערה:

הערה:

הערה:

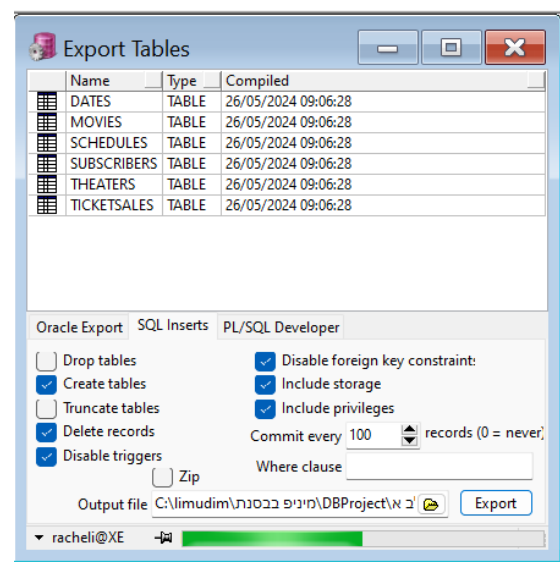
הערה:

הערה:

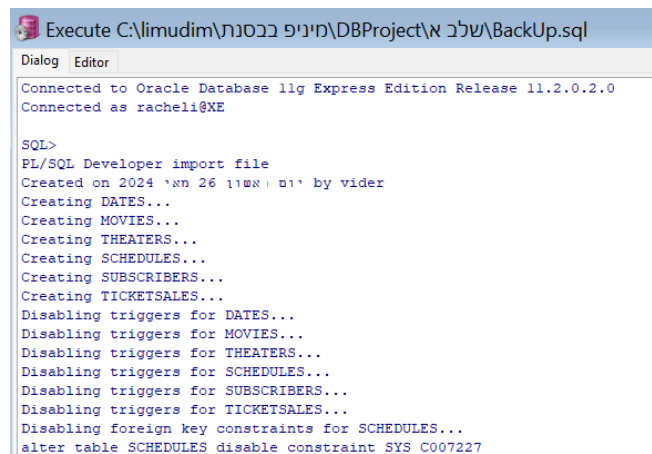
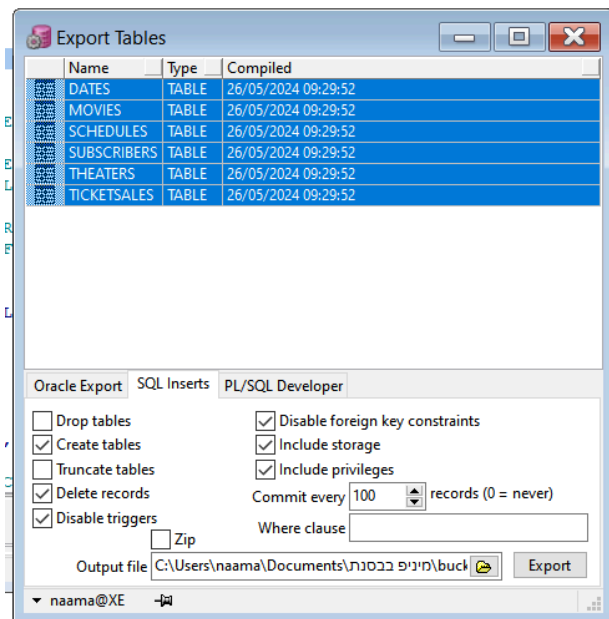
הערה:

גיבוי ושחזור:

גיבוי:



שחזור:



שלב ב

שאלות select ללא פרמטרים:
1.

תיאור השאלה: מחזיר את הדירוג הממוצע של סרטים שיצאו בכל שנה יחד עם מספר הסרטים שיצאו באותה שנה.

הרצת השאלה:

SQL	Output	Statistics
<pre>SELECT EXTRACT(YEAR FROM ReleaseDate) AS ReleaseYear, COUNT(*) AS NumberOfMovies, AVG(Rating) AS AverageRating FROM Movies GROUP BY EXTRACT(YEAR FROM ReleaseDate) ORDER BY ReleaseYear;</pre>		
31:1 ▼ naama@XE 12 rows selected in 0.157 seconds (more...)		

תוצאת השאלה:

	RELEASEYEAR	NUMBEROFMOVIES	AVERAGERATING
1	2000	9	4.66666666666667
2	2001	17	6.05882352941176
3	2002	18	5
4	2003	15	4.86666666666667
5	2004	17	5.88235294117647
6	2005	16	5.9375
7	2006	21	7
8	2007	14	5.64285714285714
9	2008	8	5.5
10	2009	22	6.13636363636364
11	2010	21	5.61904761904762
12	2011	14	4.42857142857143
13	2012	18	5.44444444444444
14	2013	17	4.64705882352941
15	2014	17	5.70588235294118
16	2015	17	5.58823529411765
17	2016	24	6.29166666666667
18	2017	14	5.57142857142857
19	2018	13	5.76923076923077
20	2019	16	3.875
21	2020	17	6.58823529411765
22	2021	13	5.76923076923077
23	2022	22	6.18181818181818
24	2023	16	5.625
25	2024	5	2.6

2.

תיאור השאלתה: מחזירה את 10 הסרטים המובילים בסך ההכנסות, כולל מספר הכרטיסים שנמכרו, מסודר לפי סדר הכנסות יורד.
הרצת השאלתה:

```

SQL      Output  Statistics
SELECT Title, TicketsSold, TotalRevenue
FROM (
  SELECT m.Title,
         COUNT(ts.TicketID) AS TicketsSold,
         SUM(ts.Price) AS TotalRevenue
  FROM TicketSales ts
  JOIN Schedules s ON ts.ScheduleID = s.ScheduleID
  JOIN Movies m ON s.MovieID = m.MovieID
  WHERE ts.IsSold = 'Y'
  GROUP BY m.Title
  ORDER BY TotalRevenue DESC
)
WHERE ROWNUM <= 10;

```

naama@XE 7 rows selected in 0.032 seconds (more...)

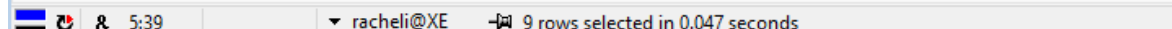
תוצאת השאלתה:

		TITLE	TICKETSSOLD	TOTALREVENUE
▶	1	Gandhi	19	870
	2	Trumbo	14	640
	3	Hancock	13	610
	4	Boyhood	12	520
	5	Watchmen	8	440
	6	Gigi	7	410
	7	Frida	7	350

3.

תיאור השאילתה: מציג את הסרטים המוקרנים היום לצד השעה, הג'אנר ומספר המקומות הפנויים.

הרצת השאילתה:

SQL	Output	Statistics
<pre>SELECT m.Title AS MovieTitle, m.Genre, d.HourH AS ScreeningTime, s.AvailableSeats FROM Movies m JOIN Schedules s ON m.MovieID = s.MovieID JOIN Dates d ON s.DateID = d.DateID WHERE TRUNC(d.DateD) = TRUNC(SYSDATE);</pre>		
		

תוצאות השאילתה:

		MOVIETITLE	GENRE	SCREENINGTIME	AVAILABLESEATS
▶	1	Witness	Sport	14:00	142
	2	Frida	Drama	20:00	216
	3	Unforgiven	Horror	23:00	294
	4	Deadpool	Sport	20:00	168
	5	Frida	Animation	23:00	95
	6	Labyrinth	Fantasy	23:00	284
	7	Gandhi	Biography	14:00	55
	8	Twister	Thriller	20:00	146
	9	Capote	Mystery	17:00	70

4.

תיאור השאילתה: השאילתה מספקת מידע לגבי מספר הכרטיסים שנמכרו לפי שנה וחודש.

הרצת השאילתה:

```

SQL Window - Queries.sql
SQL Output Statistics
SELECT
    EXTRACT(YEAR FROM D.DateD) AS Year,
    EXTRACT(MONTH FROM D.DateD) AS Month,
    COUNT(TS.TicketID) AS TicketsSold
FROM
    TicketSales TS
JOIN
    Schedules S ON TS.ScheduleID = S.ScheduleID
JOIN
    Dates D ON S.DateID = D.DateID
JOIN
    Theaters TH ON S.TheaterID = TH.TheaterID
WHERE
    TS.IsSold = 'Y'
GROUP BY
    EXTRACT(YEAR FROM D.DateD),
    EXTRACT(MONTH FROM D.DateD)
ORDER BY
    Year,
    Month;
4 rows selected in 0.031 seconds

```

תוצאות השאילתה:

	YEAR	MONTH	TICKETSSOLD
1	2024	6	130
2	2024	7	93
3	2024	8	97
4	2024	9	67

שאילות: delete

1.

תיאור השאילתה: השאילתה מוחקת את הכרטיסים שהתאריך שלהם עבר ולא נמכרו.

הרצת השאילתה:

```

SQL Window
SQL Output Statistics
DELETE FROM TicketSales
WHERE IsSold = 'N'
AND ScheduleID IN (
    SELECT ScheduleID
    FROM Schedules
    WHERE DateID IN (
        SELECT DateID
        FROM Dates
        WHERE DateD < SYSDATE
    )
);
7 rows deleted in 0.015 seconds

```

לפני:

	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
1	100	30.00	N	413	407
2	101	50.00	N	343	400
3	102	30.00	N	347	129
4	103	30.00	N	344	309
5	104	30.00	N	295	342
6	105	50.00	N	317	233
7	106	30.00	N	310	432
8	107	30.00	N	262	194
9	108	30.00	N	227	363
10	109	90.00	N	305	158
11	110	30.00	N	251	470
12	111	50.00	N	414	243
13	112	90.00	N	239	270
14	113	30.00	N	248	346
15	114	30.00	N	311	317
16	115	30.00	N	332	400
17	116	30.00	N	378	259
18	118	30.00	N	207	336
19	119	50.00	N	348	184
20	120	90.00	N	403	298
21	121	90.00	N	327	212
22	122	30.00	N	312	146
23	123	30.00	N	263	473
24	124	90.00	N	463	303
25	125	30.00	N	133	306
26	126	30.00	N	321	403
27	127	30.00	N	425	468
28	128	30.00	N	173	312
29	129	30.00	N	187	157
30	130	30.00	N	334	169
31	131	50.00	N	399	276


אחרי: ניתן לראות שה TicketSales ה121,122 נמחקו

	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
1	100	30.00	N	413	407
2	101	50.00	N	343	400
3	102	30.00	N	347	129
4	103	30.00	N	344	309
5	104	30.00	N	295	342
6	105	50.00	N	317	233
7	106	30.00	N	310	432
8	107	30.00	N	262	194
9	108	30.00	N	227	363
10	109	90.00	N	305	158
11	110	30.00	N	251	470
12	111	50.00	N	414	243
13	112	90.00	N	239	270
14	113	30.00	N	248	346
15	114	30.00	N	311	317
16	115	30.00	N	332	400
17	116	30.00	N	378	259
18	118	30.00	N	207	336
19	120	90.00	N	403	298
20	123	30.00	N	263	473
21	124	90.00	N	463	303
22	125	30.00	N	133	306
23	126	30.00	N	321	403
24	127	30.00	N	425	468
25	128	30.00	N	173	312
26	129	30.00	N	187	157
27	130	30.00	N	334	169
28	131	50.00	N	399	276
29	132	30.00	N	485	259
30	133	30.00	N	269	103
31	136	30.00	N	368	153
32	138	30.00	N	429	394

2.

תיאור השאילתה: השאילתה מוחקת כרטיסים שלא נמכרו ושייכים למנויים שלא ביצעו קנייה ב- 3 ימים האחרונים. (אותו רעיון כמו שמרוקנים את הסל באתר כשלא התבצעה תנועה בחשבון)

הרצת השאילתה:

SQL	Output	Statistics
<pre>DELETE FROM TicketSales WHERE IsSold = 'N' AND SubscriberID NOT IN (SELECT DISTINCT tsl.SubscriberID FROM TicketSales tsl JOIN Schedules s ON tsl.ScheduleID = s.ScheduleID JOIN Dates d ON s.DateID = d.DateID WHERE tsl.IsSold = 'Y' AND d.DateD >= SYSDATE - 3);</pre>		
 & 10:3 ▼ rachel@XE 23 rows deleted in 0.015 seconds		

לפני:

	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
1	100	30.00	N	413	407
2	101	50.00	N	343	400
3	102	30.00	N	347	129
4	103	30.00	N	344	309
5	104	30.00	N	295	342
6	105	50.00	N	317	233
7	106	30.00	N	310	432
8	107	30.00	N	262	194
9	108	30.00	N	227	363
10	109	90.00	N	305	158
11	110	30.00	N	251	470
12	111	50.00	N	414	243
13	112	90.00	N	239	270
14	113	30.00	N	248	346
15	114	30.00	N	311	317
16	115	30.00	N	332	400
17	116	30.00	N	378	259
18	118	30.00	N	207	336
19	119	50.00	N	348	184
20	120	90.00	N	403	298
21	121	90.00	N	327	212
22	122	30.00	N	312	146
23	123	30.00	N	263	473
24	124	90.00	N	463	303
25	125	30.00	N	133	386
26	126	30.00	N	321	403
27	127	30.00	N	425	468
28	128	30.00	N	173	312
29	129	30.00	N	187	157
30	130	30.00	N	334	169
31	131	50.00	N	399	276
32	132	30.00	N	485	259
33	133	30.00	N	269	103
34	134	50.00	N	327	297
35	136	30.00	N	368	153
36	137	30.00	N	394	362
37	138	30.00	N	429	394
38	139	30.00	N	214	130
39	140	30.00	N	279	248
40	141	30.00	N	285	417
41	142	30.00	N	393	204
42	144	50.00	N	438	422
43	145	30.00	N	447	463
44	146	30.00	N	278	181
45	147	90.00	N	312	412

אחרי: ניתן לראות שהTicketSales ה105,101,103 וכו' נמחקו.

	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
1	102	30.00	N	347	129
2	104	30.00	N	295	342
3	106	30.00	N	310	432
4	108	30.00	N	227	363
5	109	90.00	N	305	158
6	110	30.00	N	251	470
7	111	50.00	N	414	243
8	116	30.00	N	378	259
9	118	30.00	N	207	336
10	119	50.00	N	348	184
11	122	30.00	N	312	146
12	123	30.00	N	263	473
13	128	30.00	N	173	312
14	129	30.00	N	187	157
15	130	30.00	N	334	169
16	131	50.00	N	399	276
17	132	30.00	N	485	259
18	136	30.00	N	368	153
19	138	30.00	N	429	394
20	141	30.00	N	285	417
21	142	30.00	N	393	204
22	144	50.00	N	438	422
23	147	90.00	N	312	412
24	148	90.00	N	309	348
25	149	50.00	N	444	391
26	151	30.00	Y	444	368
27	152	90.00	Y	465	473
28	153	30.00	Y	374	354
29	154	30.00	Y	396	251
30	155	30.00	Y	430	353
31	156	90.00	Y	472	189
32	157	30.00	Y	341	147
33	158	30.00	Y	328	376
34	159	50.00	Y	483	500
35	161	30.00	Y	334	183
36	162	30.00	Y	128	279
37	163	30.00	Y	236	442
38	164	30.00	Y	496	164
39	165	30.00	Y	100	464
40	166	30.00	Y	287	191
41	167	30.00	Y	219	294
42	168	50.00	Y	266	320
43	169	50.00	Y	219	358
44	170	30.00	Y	449	498
45	171	50.00	Y	130	423

שאלות update:

- מעדכנת את מספר המקומות הפנויים בהקרנת סרט מסויים לפי תכולת האולם ומספר הכרטיסים שנקנו

```

SQL Window - Queries.sql
SQL Output Statistics
UPDATE Schedules s
SET AvailableSeats = (
    SELECT t.Capacity - COUNT(ts.TicketID)
    FROM Theaters t
    JOIN Schedules s2 ON t.TheaterID = s2.TheaterID
    LEFT JOIN TicketSales ts ON s2.ScheduleID = ts.ScheduleID AND ts.IsSold = 'Y'
    WHERE s2.ScheduleID = s.ScheduleID
    GROUP BY t.Capacity
);

```

84:20 racheli@XE 483 rows updated in 0.094 seconds

	SCHEDULEID	THEATERID	AVAILABLESEATS	MOVIEID	DATEID
1	100	100	241	382	563
2	101	116	69	424	383
3	102	117	287	230	417
4	103	102	166	375	321
5	105	115	57	440	483
6	106	100	242	109	192
7	108	119	199	437	197
8	109	116	69	115	576
9	110	113	172	305	183
10	111	115	57	392	171
11	112	111	277	321	433
12	113	115	56	296	214
13	115	114	187	230	113
14	116	108	151	352	503
15	117	109	216	224	217
16	118	126	71	454	184
17	119	123	164	410	127
18	120	109	216	441	208
19	121	106	138	126	362
20	122	127	121	477	504
21	123	110	146	459	305
22	124	125	50	363	558
23	125	121	58	181	558
24	126	116	70	108	114
25	127	122	142	348	104
26	128	115	55	325	552
27	129	100	240	294	112
28	130	100	237	441	118
29	131	102	163	265	551
30	132	120	282	346	280
31	133	104	262	275	384
32	134	124	241	260	198
33	135	115	55	386	514
34	136	114	187	427	434
35	137	100	241	413	261
36	138	121	57	229	113
37	139	104	263	255	559
38	140	101	193	332	135
39	141	125	51	451	435
40	142	107	33	254	426

2. מעדכנת את מחירי ההקרנות עבור הקרנות עם סרטים בעלי דירוג נמוך ושנקנו ממנו פחות מ 10 כרטיסים

```

SQL Window - Queries.sql
SQL Output Statistics
UPDATE TicketSales ts
SET Price = Price * 0.8 -- Applying a 20% discount
WHERE ScheduleID IN (
    SELECT s.ScheduleID
    FROM Schedules s
    JOIN Movies m ON s.MovieID = m.MovieID
    LEFT JOIN TicketSales ts2 ON s.ScheduleID = ts2.ScheduleID AND ts2.IsSold = 'Y'
    GROUP BY s.ScheduleID, m.Rating
    HAVING COUNT(ts2.TicketID) < 10 AND m.Rating < 5
);

```

107:1 rachel@XE 141 rows updated in 0.047 seconds

	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
1	102	30.00	N	347	129
2	104	30.00	N	295	342
3	106	30.00	N	310	432
4	108	30.00	N	227	363
5	109	90.00	N	305	158
6	110	30.00	N	251	470
7	111	50.00	N	414	243
8	116	30.00	N	378	259
9	118	30.00	N	207	336
10	119	50.00	N	348	184
11	122	30.00	N	312	146
12	123	30.00	N	263	473
13	128	30.00	N	173	312
14	129	30.00	N	187	157
15	130	30.00	N	334	169
16	131	50.00	N	399	276
17	132	30.00	N	485	259
18	136	30.00	N	368	153
19	138	30.00	N	429	394
20	141	30.00	N	285	417
21	142	30.00	N	393	204
22	144	50.00	N	438	422
23	147	90.00	N	312	412
24	148	90.00	N	309	348
25	149	50.00	N	444	391
26	151	30.00	Y	444	368
27	152	90.00	Y	465	473
28	153	30.00	Y	374	354
29	154	30.00	Y	396	251
30	155	30.00	Y	430	353
31	156	90.00	Y	472	189
32	157	30.00	Y	341	147
33	158	30.00	Y	328	376
34	159	50.00	Y	483	500
35	161	30.00	Y	334	183
36	162	30.00	Y	128	279
37	163	30.00	Y	236	442
38	164	30.00	Y	496	164
39	165	30.00	Y	100	464
40	166	30.00	Y	287	191
41	167	30.00	Y	219	294
42	168	50.00	Y	266	320
43	169	50.00	Y	219	358
44	170	30.00	Y	449	498
45	171	50.00	Y	130	423

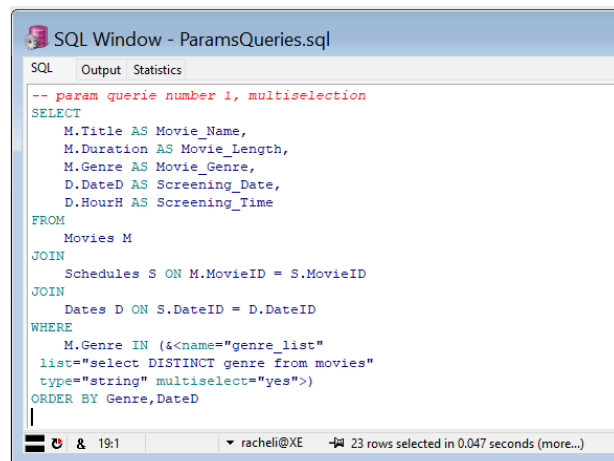
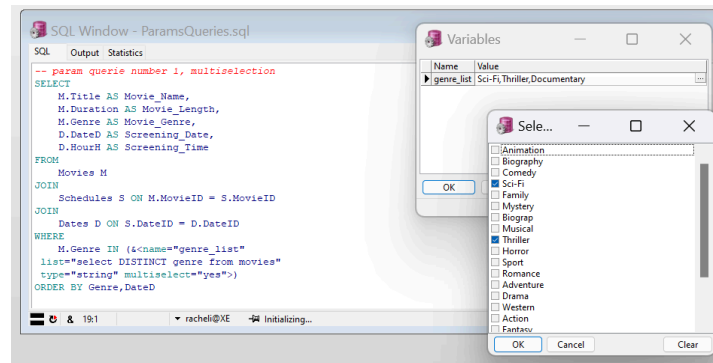
	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
1	102	30.00	N	347	129
2	104	30.00	N	295	342
3	106	30.00	N	310	432
4	108	30.00	N	227	363
5	109	72.00	N	305	158
6	110	30.00	N	251	470
7	111	40.00	N	414	243
8	116	24.00	N	378	259
9	118	24.00	N	207	336
10	123	24.00	N	263	473
11	128	24.00	N	173	312
12	129	30.00	N	187	157
13	130	30.00	N	334	169
14	131	40.00	N	399	276
15	132	30.00	N	485	259
16	136	30.00	N	368	153
17	138	30.00	N	429	394
18	141	24.00	N	285	417
19	142	24.00	N	393	204
20	144	50.00	N	438	422
21	148	72.00	N	309	348
22	151	30.00	Y	444	368
23	152	72.00	Y	465	473
24	153	30.00	Y	374	354
25	154	24.00	Y	396	251
26	155	30.00	Y	430	353
27	156	72.00	Y	472	189
28	157	24.00	Y	341	147
29	158	30.00	Y	328	376
30	159	50.00	Y	483	500
31	161	30.00	Y	334	183
32	162	24.00	Y	128	279
33	163	30.00	Y	236	442
34	164	30.00	Y	496	164
35	165	30.00	Y	100	464
36	166	30.00	Y	287	191
37	167	30.00	Y	219	294
38	168	40.00	Y	266	320
39	169	50.00	Y	219	358
40	170	30.00	Y	449	498

שאלות Select עם פרמטרים:

1.

תיאור השאלתה: המשתמש בוחר ג'אנר אחד או יותר, והשאלתה מחזירה פרטים על הסרטים בג'אנר הנבחר, לכל סרט מחזירה את: שם הסרט, זמן הסרט, ג'אנר הסרט, תאריך ההקרנה ושעת ההקרנה של הסרט.

הרצת השאלתה:



תוצאות השאלתה:

	MOVIE_NAME	MOVIE_LENGTH	MOVIE_GENRE	SCREENING_DATE	SCREENING_TIME
1	Black Swan	114	Documentary	07/06/2024	20:00
2	Deadpool	93	Documentary	08/06/2024	20:00
3	Thor	172	Documentary	18/06/2024	17:00
4	Avatar	98	Documentary	19/06/2024	23:00
5	Holes	175	Documentary	27/06/2024	20:00
6	Selma	118	Documentary	11/07/2024	20:00
7	Whiplash	112	Documentary	14/07/2024	17:00
8	Platoon	171	Documentary	14/07/2024	23:00
9	Thor	172	Documentary	19/07/2024	23:00
10	Doubt	153	Documentary	22/07/2024	20:00
11	Holes	175	Documentary	27/07/2024	20:00
12	Whiplash	112	Documentary	04/08/2024	23:00
13	Holes	175	Documentary	21/08/2024	17:00
14	Black Swan	114	Documentary	24/08/2024	23:00
15	Casino	64	Documentary	27/08/2024	17:00
16	Casino	64	Documentary	07/09/2024	23:00
17	Rio	73	Documentary	07/09/2024	20:00
18	Black Swan	114	Documentary	24/09/2024	17:00
19	Arrival	127	Documentary	27/09/2024	20:00
20	Superbad	111	Sci-Fi	06/06/2024	20:00
21	Hancock	110	Sci-Fi	07/06/2024	14:00
22	Tombstone	122	Sci-Fi	11/06/2024	14:00
23	Hugo	90	Sci-Fi	12/06/2024	23:00
24	Hugo	90	Sci-Fi	18/06/2024	17:00
25	Trumbo	180	Sci-Fi	23/06/2024	20:00
26	Trumbo	85	Sci-Fi	28/06/2024	14:00
27	Airplane	82	Sci-Fi	30/06/2024	20:00
28	Superbad	138	Sci-Fi	09/07/2024	23:00
29	Moana	151	Sci-Fi	10/07/2024	20:00
30	Risen	163	Sci-Fi	11/07/2024	17:00
31	Trumbo	180	Sci-Fi	17/07/2024	14:00
32	Gandhi	67	Sci-Fi	22/07/2024	14:00
33	Trumbo	180	Sci-Fi	23/07/2024	23:00
34	Arrival	126	Sci-Fi	24/07/2024	20:00
35	Witness	97	Sci-Fi	27/07/2024	20:00
36	Paprika	156	Sci-Fi	03/08/2024	20:00
37	Hancock	110	Sci-Fi	06/08/2024	23:00
38	Black Swan	66	Sci-Fi	07/08/2024	14:00
39	Grease	135	Sci-Fi	09/08/2024	17:00

2.

תיאור השאילתה: המשתמש מכניס דירוג, ותאריך והשאילתה מחזירה את כל הסרטים בתאריך שנבחר שהדירוג שלהם גדול או שווה מהדירוג שנבחר. על כל סרט, מוצג הדירוג שלו, תאריך ושעת ההקרנה שלו.

הרצת השאילתה:

SQL Window - ParamsQueries.sql

SQL

Output

Statistics

```
-- param query number 2
SELECT
  M.Title AS Movie_Name,
  M.Rating AS Movie_Rating,
  D.DateD AS Screening_Date,
  D.HourH AS Screening_Time
FROM
  Movies M
JOIN
  Schedules S ON M.MovieID = S.MovieID
JOIN
  Dates D ON S.DateID = D.DateID
WHERE
  M.Rating >= <name="Minimum_desired_rating" hint= "Rating between 1-10" type="integer" default=5>
  AND D.DateD = <name="screening_date" type="date" required=true>
ORDER BY
  M.Rating DESC, D.HourH;
```

Variables

Name

Value

Minimum_desired_rating

4

screening_date

02/06/2024

OK

Cancel

Clear

```

SQL Window - ParamsQueries.sql
SQL Output Statistics
-- param query number 2
SELECT
  M.Title AS Movie_Name,
  M.Rating AS Movie_Rating,
  D.DateD AS Screening_Date,
  D.HourH AS Screening_Time
FROM
  Movies M
JOIN
  Schedules S ON M.MovieID = S.MovieID
JOIN
  Dates D ON S.DateID = D.DateID
WHERE
  M.Rating >= <name="Minimum_desired_rating" hint= "Rating between 1-10" type="integer" default=5>
  AND D.DateD = <name="screening_date" type="date" required=true>
ORDER BY
  M.Rating DESC, D.HourH;

```

3 rows selected in 0.047 seconds (more...)

תוצאות השאילתה:

	MOVIE_NAME	MOVIE_RATING	SCREENING_DATE	SCREENING_TIME
1	Twister	10	02/06/2024	20:00
2	Deadpool	8	02/06/2024	20:00
3	Labyrinth	7	02/06/2024	23:00

3.

תיאור השאילתה: המשתמש בוחר במאי, והשאילתה מחזירה את רשימת הסרטים על אותו במאי עם הדירוג הכי גבוה.
הרצת השאילתה:

```

SQL Window - ParamsQueries.sql
SQL Output Statistics
-- param query number 3
SELECT
  M.Director AS Director_Name,
  M.Title AS Movie_Name,
  M.Rating AS Movie_Rating,
  D.DateD AS Screening_Date,
  D.HourH AS Screening_Time
FROM
  Movies M
JOIN
  Schedules S ON M.MovieID = S.MovieID
JOIN
  Dates D ON S.DateID = D.DateID
JOIN
  Theaters T ON S.TheaterID = T.TheaterID
WHERE
  M.Director = <name="director" type="string" list="SELECT DISTINCT Director FROM Movies ORDER BY Director" required=true>
  AND M.Rating = (
    SELECT MAX(Rating)
    FROM Movies
    WHERE Director = M.Director
  )
ORDER BY
  M.Title DESC;

```

Initializing...

Variables

Name	Value
director	Aaron Mangon

OK

```

SQL Window - ParamsQueries.sql
SQL Output Statistics
-- param query number 3
SELECT
  M.Director AS Director_Name,
  M.Title AS Movie_Name,
  M.Rating AS Movie_Rating,
  D.DateD AS Screening_Date,
  D.HourH AS Screening_Time
FROM
  Movies M
JOIN
  Schedules S ON M.MovieID = S.MovieID
JOIN
  Dates D ON S.DateID = D.DateID
JOIN
  Theaters T ON S.TheaterID = T.TheaterID
WHERE
  M.Director = <name="director" type="string" list="SELECT DISTINCT Director FROM Movies ORDER BY Director" required=true>
  AND M.Rating = (
    SELECT MAX(Rating)
    FROM Movies
    WHERE Director = M.Director
  )
ORDER BY
  M.Title DESC;

```

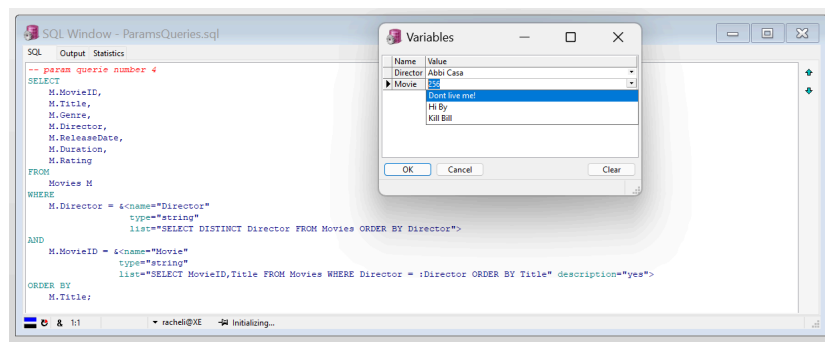
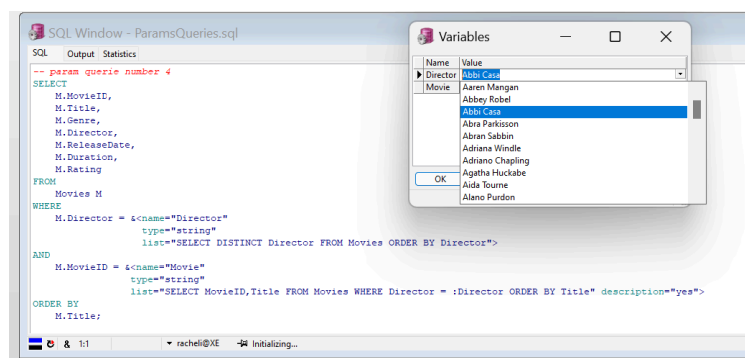
1 row selected in 0.047 seconds

תוצאות השאילתה:

	DIRECTOR_NAME	MOVIE_NAME	MOVIE_RATING	SCREENING_DATE	SCREENING_TIME
1	Aaren Mangan	Black Swan	4	19/09/2024	23:00

4.

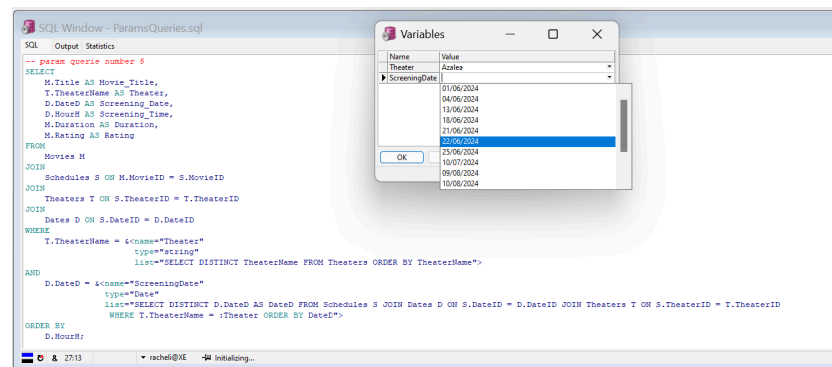
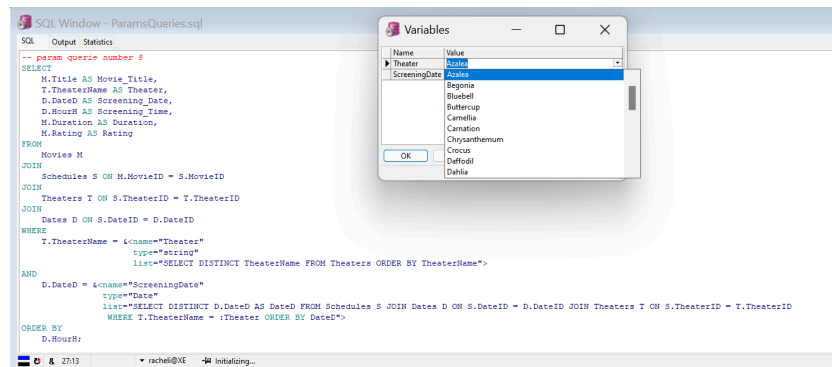
תיאור השאילתה: המשתמש בוחר שם של במאי ואז לפי הבמאי שבחר מוצעת לו רשימת הסרטים של אותו הבמאי, ומתוך הרשימה הזו המשתמש בוחר סרט ועל כל סרט מוצג לו: הרצת השאילתה:



תוצאות השאילתה:

	MOVIEID	TITLE	GENRE	DIRECTOR	RELEASEDATE	DURATION	RATING
1	456	Kill Bill	Western	Abbi Casa	04/05/2004	137	7

5. תיאור השאילתה: המשתמש בוחר אולם ותאריך, והשאילתה מחזירה את רשימת הסרטים המוקרנים באותו אולם ובאותו תאריך, על כל סרט מוצג: שם הסרט, שם הקולנוע, תאריך ההקרנה, זמן ההקרנה, משך הסרט ודירוג הסרט.
הרצת השאילתה:



תוצאות השאילתה:

	MOVIE_TITLE	THEATER	SCREENING_DATE	SCREENING_TIME	DURATION	RATING
1	Gravity	Azalea	22/06/2024	14:00	63	5

אילוצים:

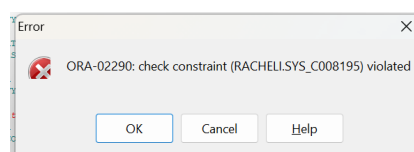
1. האילוע: משך הסרט הוא לכל היותר 3 שעות (180 דק').

```
-- MOVIES Table  
ALTER TABLE Movies  
MODIFY Duration INT CHECK (Duration <= 180);
```

ניסיון להפרת האילוע:

```
INSERT INTO Movies (MovieID, Title, Genre, Director, ReleaseDate, Duration, Rating)  
VALUES (1, 'Test Movie', 'Genre', 'Director', DATE '2023-01-01', 200, 5);
```

שגיאה:



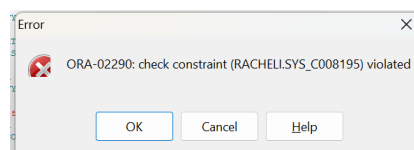
2. האילוע: הדירוג של סרט הוא בין 1 ל-10.

```
ALTER TABLE Movies  
MODIFY Rating INT CHECK (Rating >= 1 AND Rating <= 10);
```

ניסיון להפרת האילוע:

```
INSERT INTO Movies (MovieID, Title, Genre, Director, ReleaseDate, Duration, Rating)  
VALUES (2, 'Test Movie 2', 'Genre', 'Director', DATE '2023-01-01', 120, 11);
```

שגיאה:



3. האילוע: הצירוף של שעה, יום, ותאריך הם UNIQUE - הצירוף לא יחזור על עצמו בורה נוספת בטבלה.

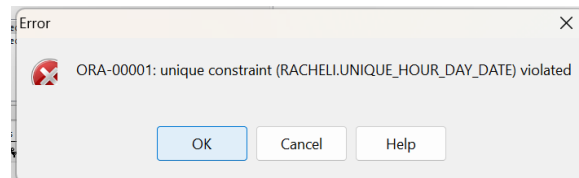
```
-- DATES Table  
ALTER TABLE Dates  
ADD CONSTRAINT unique_hour_day_date UNIQUE (HourH, DayD, DateD);
```

ניסיון להפרת האילוע:

```
INSERT INTO Dates (DateID, HourH, DayD, DateD)
VALUES (1, '10:00', 'Monday', DATE '2023-01-01');

INSERT INTO Dates (DateID, HourH, DayD, DateD)
VALUES (2, '10:00', 'Monday', DATE '2023-01-01');
```

שגיאה:



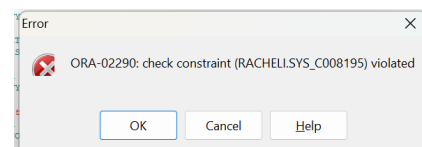
4. האילוח: מספר המושבים בכל אולם קולנוע הוא חיובי.

```
-----
ALTER TABLE Theaters
MODIFY Capacity INT CHECK (Capacity >= 0);
```

ניסיון להפרת האילוח:

```
INSERT INTO Theaters (TheaterID, TheaterName, Capacity)
VALUES (1, 'Theater 1', -10);
```

שגיאה:



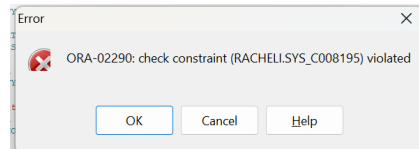
5. האילוח: מספר המקומות הפנויים הוא 0 בהתחלה ובדק שמכניס מספר חיובי (או 0).

```
ALTER TABLE Schedules
MODIFY AvailableSeats INT DEFAULT 0 CHECK (AvailableSeats >= 0);
```

ניסיון להפרת האילוח:

```
INSERT INTO Schedules (ScheduleID, TheaterID, AvailableSeats, MovieID, DateID)
VALUES (1, 1, -5, 1, 1);
```

שגיאה:



6. האילוח: התאריך והאולם בהם נקבעה הקרנת סרט הם ייחודיים. (כדי למנוע התנגשות של סרטים באותו אולם באותה שעה)

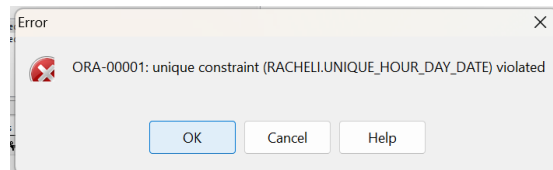
```
ALTER TABLE Schedules
ADD CONSTRAINT unique_date_theater UNIQUE(DateID, TheaterID);
```

ניסיון להפרת האילוח:

```
INSERT INTO Schedules (ScheduleID, TheaterID, AvailableSeats, MovieID, DateID)
VALUES (2, 1, 50, 1, 1);

INSERT INTO Schedules (ScheduleID, TheaterID, AvailableSeats, MovieID, DateID)
VALUES (3, 1, 50, 1, 1);
```

שגיאה:



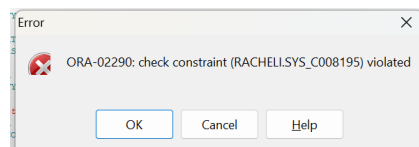
7. האילוח: אחוז ההנחה הדיפולטיבי הוא 30, והוא יכול להיות מאותחל ל 10\20\30\40\50.

```
ALTER TABLE Subscribers
MODIFY DiscountPercents INT DEFAULT 30 CHECK(DiscountPercents IN (10, 20, 30, 40, 50));
```

ניסיון להפרת האילוח:

```
INSERT INTO Subscribers (SubscriberID, DiscountPercents, ExpirationDate)
VALUES (1, 15, DATE '2023-01-01');
```

שגיאה:



8. האילוח: המחיר הבסיסי של סרט יכול להיות בין 30\50\90.


```

-- RENAME TABLE
ALTER TABLE TicketSales
MODIFY Price NUMERIC(5, 2) CHECK(Price IN (30, 50, 90));

```

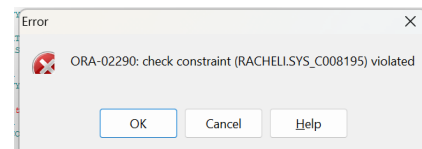
ניסיון להפרת האילוץ:

```

INSERT INTO TicketSales (TicketID, Price, IsSold, ScheduleID, SubscriberID)
VALUES (1, 35, 'N', 1, NULL);

```

שגיאה:



9. האילוץ: התז של המנויי הוא נאל בדיפולט כי באופן דיפולטיבי בן אדם רנדומלי קונה את הכרטיס ולא מנויי.

```

ALTER TABLE TicketSales
MODIFY SubscriberID NUMERIC(3) DEFAULT NULL;

```

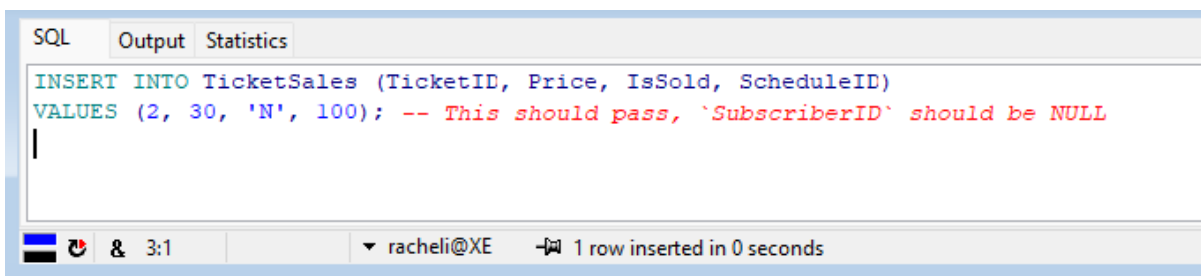
ניסיון לבדיקת האילוץ:

```

INSERT INTO TicketSales (TicketID, Price, IsSold, ScheduleID)
VALUES (2, 30, 'N', 100); -- This should pass, `SubscriberID` should be NULL

```

הבדיקה רצה בהצלחה:



פונקציות:

1. Find_Available_Theaters

תיאור מילולי: הפונקציה מקבלת מזהה תאריך (תאריך ושעה מסויים) ובודקת איזה אולמות פנויים בתאריך ובשעה הרצויה, הפונקציה מחזירה Cursor ובו ID של כל האולמות הפנויים בזמן זה.

הקוד:

```
1 CREATE OR REPLACE FUNCTION Find_Available_Theaters(p_DateID NUMBER)
2   RETURN SYS_REFCURSOR
3   IS
4
5   -- Ref Cursor variable to return result set
6   rc SYS_REFCURSOR;
7
8   -- Variables for error handling
9   v_error_code NUMBER;
10  v_error_msg VARCHAR2(4000);
11
12 BEGIN
13   -- Initialize ref cursor
14   OPEN rc FOR
15     SELECT TheaterID
16     FROM Theaters
17     WHERE TheaterID NOT IN (
18       SELECT TheaterID
19       FROM Schedules
20       WHERE DateID = p_DateID
21     );
22
23   -- Return the ref cursor
24   RETURN rc;
25
26 EXCEPTION
27   WHEN NO_DATA_FOUND THEN
28     -- Handle no data found error
29     v_error_code := SQLCODE;
30     v_error_msg := 'No available theaters found for DateID ' || p_DateID;
31     DBMS_OUTPUT.PUT_LINE(v_error_msg);
32     CLOSE rc;
33     RETURN NULL;
34
35   WHEN OTHERS THEN
36     -- Handle other exceptions
37     v_error_code := SQLCODE;
38     v_error_msg := 'Error in FindAvailableTheaters function: ' || SQLERRM;
39     DBMS_OUTPUT.PUT_LINE(v_error_msg);
40     CLOSE rc;
41     RETURN NULL;
42 END FindAvailableTheaters;
```

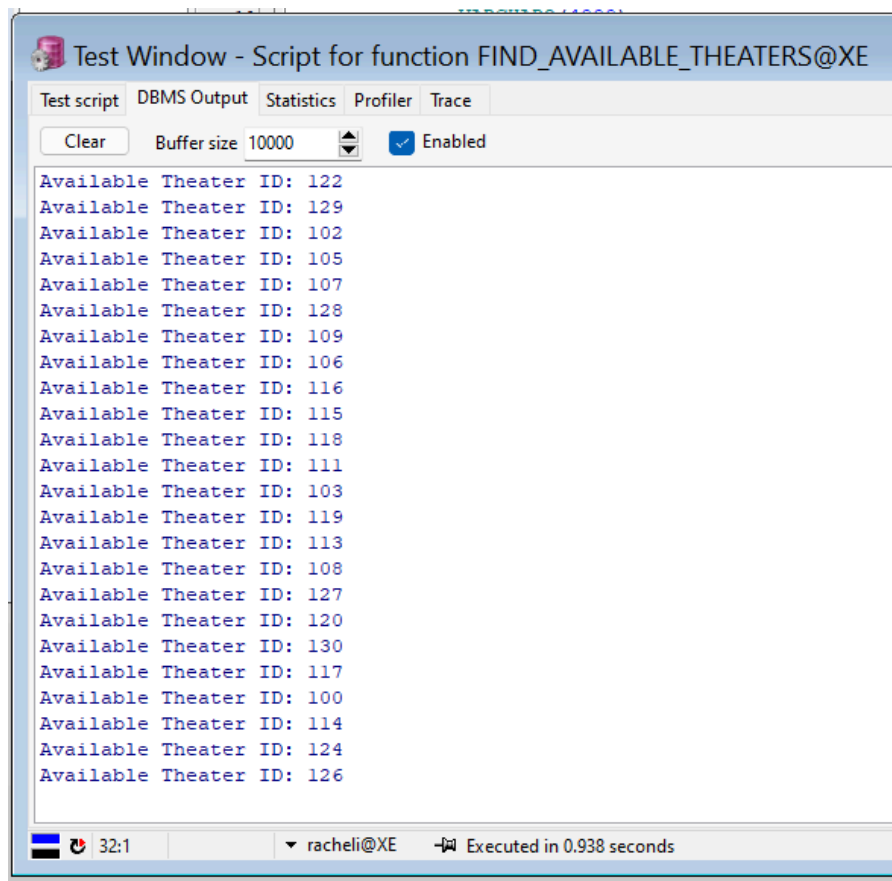
בדיקת הפונקציה:

```
1 DECLARE
2   v_DateID NUMBER := 150; -- Replace with the desired DateID
3   v_rc SYS_REFCURSOR;
4   v_TheaterID Theaters.TheaterID%TYPE;
5 BEGIN
6   -- Call the function and get the REF CURSOR
7   v_rc := Find_Available_Theaters(v_DateID);
8
9   -- Fetch the results from the cursor and print them
10  LOOP
11    FETCH v_rc INTO v_TheaterID;
12    EXIT WHEN v_rc%NOTFOUND;
13    DBMS_OUTPUT.PUT_LINE('Available Theater ID: ' || v_TheaterID);
14  END LOOP;
15
16  -- Close the cursor
17  CLOSE v_rc;
18 EXCEPTION
19  WHEN OTHERS THEN
20    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
21    IF v_rc%ISOPEN THEN
22      CLOSE v_rc;
23    END IF;
24 END;
```

1:1 | racheli@XE | Executed in 0.938 seconds

התוצאה:

ניתן לראות שלמעט חדרים בודדים כמעט כל חדרי הקולנוע היו פנויים בזמן זה



2. GET_AVAILABLE_SCHEDULES

תיאור מילולי: הפונקציה מקבלת מזהה סרט, ובודקת איזה הקרנות קיימות לסרט הזה ובאילו מהם יש מקומות פנויים, הפונקציה מחזירה Cursor ובו כל מזה"י ההקרנות שהיא מצאה

הקוד:

```

1 CREATE OR REPLACE FUNCTION GET_AVAILABLE_SCHEDULES(p_movie_id IN Schedules.MovieID%TYPE)
2 RETURN SYS_REFCURSOR
3 IS
4     v_ref_cursor SYS_REFCURSOR;
5 BEGIN
6     OPEN v_ref_cursor FOR
7     SELECT ScheduleID FROM Schedules
8     WHERE MovieID = p_movie_id
9     AND AvailableSeats > 0;
10
11     RETURN v_ref_cursor;
12 END GET_AVAILABLE_SCHEDULES;

```

בדיקת הפונקציה:

```

1 DECLARE
2   v_movie_id Schedules.MovieID%TYPE := 265; -- Replace with the desired MovieID
3   v_ref_cursor SYS_REFCURSOR;
4   v_ScheduleID Schedules.ScheduleID%TYPE;
5 BEGIN
6   -- Call the function and get the REF CURSOR
7   v_ref_cursor := GET_AVAILABLE_SCHEDULES(v_movie_id);
8
9   -- Fetch the results from the cursor and print them
10  LOOP
11    FETCH v_ref_cursor INTO v_ScheduleID;
12    EXIT WHEN v_ref_cursor%NOTFOUND;
13    DBMS_OUTPUT.PUT_LINE('Available Schedule ID: ' || v_ScheduleID);
14  END LOOP;
15
16  -- Close the cursor
17  CLOSE v_ref_cursor;
18 EXCEPTION
19  WHEN OTHERS THEN
20    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
21    IF v_ref_cursor%ISOPEN THEN
22      CLOSE v_ref_cursor;
23    END IF;
24 END;

```

1:1 0:03 racheli@XE Executed in 3.828 seconds

התוצאה:

עבור מזהה הסרט 265 הפונקציה החזירה 7 הקרנות של הסרט בהם יש מקום פנוי

```

Available Schedule ID: 131
Available Schedule ID: 220
Available Schedule ID: 277
Available Schedule ID: 300
Available Schedule ID: 333
Available Schedule ID: 372
Available Schedule ID: 571

```

8:1 0:03 racheli@XE Executed in 3.828 seconds

3. GetMovieWithMaxSchedules

תיאור מילולי: הפונקציה מחפשת את הסרט עם מספר ההקרנות המקסימלי ומחזירה רשומה של הסרט (אם יש כמה סרטים עם מספר הקרנות גבוה ביותר הפונקציה תחזיר אחד מחר באופן שרירותי)

הקוד:

```

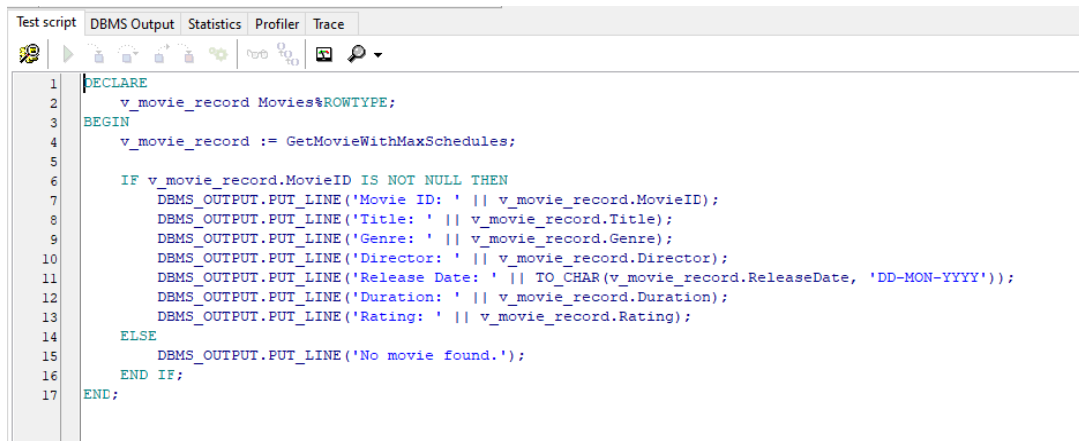
CREATE OR REPLACE FUNCTION GetMovieWithMaxSchedules
RETURN Movies%ROWTYPE
IS
    v_movie_record Movies%ROWTYPE;
BEGIN
    SELECT m.*
    INTO v_movie_record
    FROM Movies m
    WHERE m.MovieID = (
        SELECT MovieID
        FROM (
            SELECT s.MovieID
            FROM Schedules s
            GROUP BY s.MovieID
            ORDER BY COUNT(s.ScheduleID) DESC
        )
        WHERE ROWNUM = 1
    );

    RETURN v_movie_record;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        -- If no data is found, handle accordingly
        RETURN NULL;
    WHEN OTHERS THEN
        -- Handle any other exceptions
        RAISE;
END GetMovieWithMaxSchedules;

```

בדיקת הפונקציה:



```

1 DECLARE
2     v_movie_record Movies%ROWTYPE;
3 BEGIN
4     v_movie_record := GetMovieWithMaxSchedules;
5
6     IF v_movie_record.MovieID IS NOT NULL THEN
7         DBMS_OUTPUT.PUT_LINE('Movie ID: ' || v_movie_record.MovieID);
8         DBMS_OUTPUT.PUT_LINE('Title: ' || v_movie_record.Title);
9         DBMS_OUTPUT.PUT_LINE('Genre: ' || v_movie_record.Genre);
10        DBMS_OUTPUT.PUT_LINE('Director: ' || v_movie_record.Director);
11        DBMS_OUTPUT.PUT_LINE('Release Date: ' || TO_CHAR(v_movie_record.ReleaseDate, 'DD-MON-YYYY'));
12        DBMS_OUTPUT.PUT_LINE('Duration: ' || v_movie_record.Duration);
13        DBMS_OUTPUT.PUT_LINE('Rating: ' || v_movie_record.Rating);
14    ELSE
15        DBMS_OUTPUT.PUT_LINE('No movie found.');

```

התוצאה:

הפונקציה החזירה רשומה בה כל פרטי הסרט

×

Test scriptDBMS OutputStatisticsProfilerTrace

ClearBuffer size10000☐ Enabled

Movie ID: 123
Title: Twister
Genre: Horror
Director: Ursola Masson
Release Date: 27-7-2022
Duration: 64
Rating: 6
|

פרוצדורות:

1. ADD_SCHEDULE

תיאור מילולי: הפרוצדורה מוסיפה הקרנת סרט חדשה לטבלת לוחות הזמנים. הפרוצדורה מקבלת מזהה סרט, מזהה תאריך ו cursor מוכן מראש של אולמות קולנוע פנויים ומקצה את אולם הקולנוע הראשון שזמין לסרט המבוקש בתאריך הנתון.

הקוד:

```
CREATE OR REPLACE PROCEDURE ADD_SCHEDULE (
    p_date_id IN Dates.dateID%TYPE,
    p_movie_id IN Movies.MovieID%TYPE,
    p_theater_cursor IN OUT SYS_REFCURSOR
) IS
    v_new_schedule_id Schedules.ScheduleID%TYPE;
    v_theater_id Theaters.TheaterID%TYPE; -- Variable to hold TheaterID
    v_availableseats Schedules.Availableseats%TYPE;
BEGIN

    -- Calculate the new scheduleID
    SELECT NVL(MAX(ScheduleID), 0) + 1
    INTO v_new_schedule_id
    FROM Schedules;

    -- Fetch the first TheaterID from the cursor
    FETCH p_theater_cursor INTO v_theater_id;

    -- Calculate the Availableseats
    SELECT t.Capacity INTO v_availableseats
    FROM Theaters t
    WHERE t.TheaterID = v_theater_id;

    -- Insert new schedule into Schedules table
    INSERT INTO Schedules (ScheduleID, DateID, MovieID, TheaterID, Availableseats)
    VALUES (v_new_schedule_id, p_date_id, p_movie_id, v_theater_id, v_availableseats);

    -- Commit the transaction
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('New Schedule ID created: ' || v_new_schedule_id);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No data found in the cursor.');
```

בסיס הנתונים לפני:

	SCHEDULEID	THEATERID	AVAILABLESEATS	MOVIEID	DATEID
464	580	102	167	101	452
465	581	123	165	272	157
466	582	113	172	483	125
467	583	113	172	230	572
468	584	104	263	129	122
469	585	105	168	252	579
470	586	116	70	301	105
471	587	123	165	425	339
472	588	130	94	361	368
473	589	125	51	297	193
474	590	107	33	151	553
475	591	118	162	464	289
476	592	102	167	185	295
477	593	118	162	371	250
478	594	112	296	461	242
479	595	124	242	343	503
480	596	109	216	305	303
481	597	129	99	242	271
482	598	128	260	447	430
483	599	121	58	347	175
484	600	121	58	123	456

בדיקת הפרוצדורה:

```

1 DECLARE
2   v_DateID NUMBER := 150; -- Replace with the desired DateID
3   v_theater_cursor SYS_REFCURSOR;
4   v_TheaterID Theaters.TheaterID%TYPE;
5 BEGIN
6   -- Call the function and get the REF CURSOR
7   v_theater_cursor := Find_Available_Theaters(v_DateID);
8   -- Call the ADD_SCHEDULE procedure with the cursor
9   ADD_SCHEDULE(
10    p_date_id => 150, -- Example DateID
11    p_movie_id => 150, -- Example MovieID
12    p_theater_cursor => v_theater_cursor
13  );
14 END;

```

1:1 racheli@XE Executed in 0.797 seconds

אחרי הרצת הביקתה קיבלנו הודעה שנוצרה הקרנה חדשה עם מזהה 601

```

Test Window - Script for procedure ADD_SCHEDULE@XE
Test script DBMS Output Statistics Profiler Trace
Clear Buffer size 10000 Enabled
New Schedule ID created: 601
2:1 racheli@XE Executed in 0.797 seconds

```

בסיס הנתונים אחרי הבדיקה:

ניתן לראות את ההקרנה 601 החדשה שנוספה

	SCHEDULEID	THEATERID	AVAILABLESEATS	MOVIEID	DATEID
462	578	124	242	497	256
463	579	109	216	403	380
464	580	102	167	101	452
465	581	123	165	272	157
466	582	113	172	483	125
467	583	113	172	230	572
468	584	104	263	129	122
469	585	105	168	252	579
470	586	116	70	301	105
471	587	123	165	425	339
472	588	130	94	361	368
473	589	125	51	297	193
474	590	107	33	151	553
475	591	118	162	464	289
476	592	102	167	185	295
477	593	118	162	371	250
478	594	112	296	461	242
479	595	124	242	343	503
480	596	109	216	305	303
481	597	129	99	242	271
482	598	128	260	447	430
483	599	121	58	347	175
484	600	121	58	123	456
485	601	121	58	150	150

2. SELL_TICKETS

תיאור מילולי: הפרוצדורה מוכרת מספר מוגדר של כרטיסים עבור הקרנה מסוימת. הפרוצדורה מקבלת מזהה של הקרנה ומספר כרטיסים למכירה היא מעדכנת את סטטוס ה-IsSold של כרטיסים זמינים ל-'Y' ומקטינה את מספר המושבים הפנויים בטבלת לוחות הזמנים.

הקוד:

```

3 CREATE OR REPLACE PROCEDURE SELL_TICKETS(p_schedule_id IN TicketSales.ScheduleID%TYPE, p_num_tickets IN NUMBER)
IS
3   CURSOR c_tickets IS
3     SELECT TicketID FROM TicketSales
      WHERE ScheduleID = p_schedule_id
      AND IsSold = 'N'
      FOR UPDATE;

  v_ticket_id TicketSales.TicketID%TYPE;
  v_available_seats Schedules.AvailableSeats%TYPE;
  v_sold_tickets NUMBER := 0; -- מספר הכרטיסים שנמכרו
  v_tickets_left NUMBER := p_num_tickets; -- מסתנה ימעקב אחר מספר הכרטיסים שנותרו למכירה
BEGIN
  OPEN c_tickets;

3  LOOP
    FETCH c_tickets INTO v_ticket_id;
    EXIT WHEN c_tickets%NOTFOUND OR v_tickets_left <= 0;

3    UPDATE TicketSales
      SET IsSold = 'Y'
      WHERE CURRENT OF c_tickets;

    v_sold_tickets := v_sold_tickets + 1;
    v_tickets_left := v_tickets_left - 1;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE( v_sold_tickets || 'Tickets where sold ' );

  CLOSE c_tickets;

3  SELECT AvailableSeats INTO v_available_seats
    FROM Schedules
   WHERE ScheduleID = p_schedule_id;

3  UPDATE Schedules
    SET AvailableSeats = v_available_seats - v_sold_tickets
   WHERE ScheduleID = p_schedule_id;

  COMMIT;
EXCEPTION
3  WHEN OTHERS THEN
    ROLLBACK;
    RAISE;
END SELL_TICKETS;|

```

בסיס הנתונים לפני:

עבור מזהה ההקרנה 327 קיימים ארבעה כרטיסים 2 מתוכם עדיין לא נמכרו

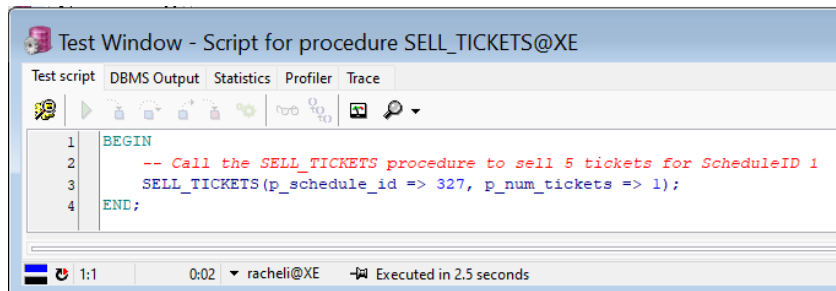
```

SELECT *
FROM TicketSales
WHERE scheduleid= 327

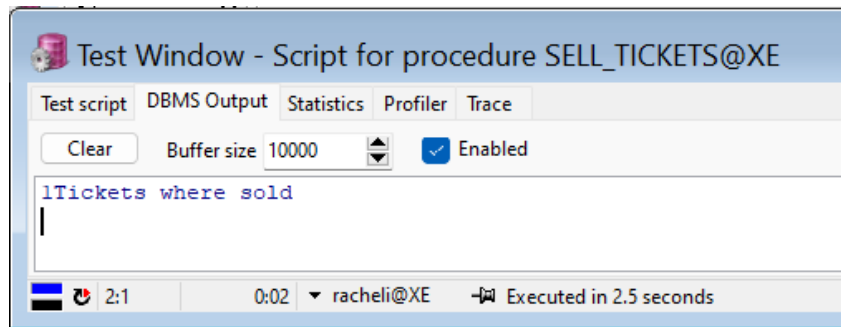
```

	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
▶	1	121	90.00 N	327	212
	2	134	50.00 N	327	297
	3	358	90.00 Y	327	223
	4	385	30.00 Y	327	451

בדיקת הפרוצדורה:



אחרי הרצת הבדיקה קיבלנו הודעה שכרטיס אחד נמכר בהצלחה



בסיס הנתונים אחרי הבדיקה:

ניתן לראות שכרטיס 121 נמכר בהצלחה!

```

SELECT *
FROM TicketSales
WHERE scheduleid= 327

```

	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
1	121	90.00	Y	327	212
2	134	50.00	N	327	297
3	358	90.00	Y	327	223
4	385	30.00	Y	327	451

3. emptyingTheader

תיאור מילולי: הפרוצדורה מרוקנת אולם מסוים בתאריך מסוים. היא מעביר הכל הקרנה לאולם אחר הפנוי באותה עת. השעה, ושאר הפרטים כמובן יישארו זהים. אם אין אולם פנוי בשעה הרצויה, היא תודיע על כך ותשאיר את ההקרנה באולם המקורי. משתמשת בפונקציית עזר Find_Available_Theaters שמוצאת אולם ריק בתאריך מסוים.

הקוד:

```

1 create or replace procedure emptyingTheader(TheaderNumber in out Theaters.TheaterID%type, datel in out Dates.DateID%type) is
2 cursor SchedulesNeedMove is select ScheduleID from Schedules where TheaterID=TheaderNumber
3 and DateID IN (select DateID from Dates where DateD = datel);
4 ScheduleNumber Schedules.ScheduleID%type;
5 TheaterNewNumber Theaters.TheaterID%type;
6 DateNumber Schedules.DateID%type;
7
8 v_theater_cursor SYS_REFCURSOR;
9
10 begin
11 open SchedulesNeedMove;
12 fetch SchedulesNeedMove into ScheduleNumber;
13 loop
14 exit when SchedulesNeedMove%notfound;
15 select DateID into DateNumber from Schedules where ScheduleID=ScheduleNumber;
16 v_theater_cursor:=Find_Available_Theaters(DateNumber);
17 if v_theater_cursor%notfound then
18 dbms_output.put_line( 'There is no empty hall at this time' );
19 else
20 -- Fetch the first TheaterID from the cursor
21 FETCH v_theater_cursor INTO TheaterNewNumber;
22 update Schedules set TheaterID=TheaterNewNumber where ScheduleID= ScheduleNumber;
23 commit;
24 dbms_output.put_line( 'new Theater ' || TheaterNewNumber);
25 end if;
26 fetch SchedulesNeedMove into ScheduleNumber;
27 end loop;
28 close SchedulesNeedMove;
29 end emptyingTheader;
30

```

בסיס הנתונים לפני:

	SCHEDULEID	THEATERID	AVAILABLESEATS	MOVIEID	DATEID
1	100	121	241	382	563
2	101	121	69	424	383
3	102	112	287	230	417
4	103	102	166	375	321
5	105	115	57	440	483
6	106	100	242	109	192
7	108	119	199	437	197
8	109	116	69	115	576
9	110	113	172	305	183
10	111	115	57	392	171
11	112	111	277	321	433
12	113	115	56	296	214
13	115	114	187	230	113
14	116	108	151	352	503
15	117	109	216	224	217
16	118	126	71	454	184
17	119	123	164	410	127
18	120	109	216	441	208

בדיקת הפרוצדורה:

Test script	DBMS Output	Statistics	Profiler	Trace
<pre> 1 begin 2 -- Call the procedure 3 emptyingtheader(theadernumber => :theadernumber, 4 datel => :date1); 5 end; </pre>				
Variable	Type	Value		
theadernumber	Integer	102		
date1	Date	26/07/2024		

בסיס הנתונים אחרי הבדיקה:

ניתן לראות שבשורה 4 הTHEATERID התחלף מ-102 ל-121

	SCHEDULEID	THEATERID	AVAILABLESEATS	MOVIEID	DATEID
1	100	121	241	382	563
2	101	121	69	424	383
3	102	112	287	230	417
4	103	121	166	375	321
5	105	115	57	440	483
6	106	100	242	109	192
7	108	119	199	437	197
8	109	116	69	115	576
9	110	113	172	305	183
10	111	115	57	392	171
11	112	111	277	321	433
12	113	115	56	296	214
13	115	114	187	230	113
14	116	108	151	352	503
15	117	109	216	224	217
16	118	126	71	454	184
17	119	123	164	410	127
18	120	109	216	441	208
19	121	106	138	126	362
20	122	127	121	477	504

תוכניות ראשיות:

1. תיאור מילולי:

התוכנית שלהלן נעזרת בפונקציה GET_AVAILABLE_SCHEDULES שכתבנו על מנת למצוא מזהי הקרנות זמינים עבור סרט מסוים ומשתמשת בפרוצדורה SELL_TICKETS שכתבנו על מנת למכור כרטיסים להקרנות אלו.

הקוד:

```
SQL Window - plsql program1.sql
SQL Output Statistics
DECLARE
  v_movie_id Movies.MovieID%TYPE;
  v_ref_cursor SYS_REFCURSOR;
  v_schedule_id Schedules.ScheduleID%TYPE;
  v_num_tickets NUMBER := &"num_of_tickets";
BEGIN
  select MOVIEID into v_movie_id from Movies where MovieID = &"name="Movie_Name" type="string" List="select MovieID, Title from Movies" description="yes" >;

  v_ref_cursor := GET_AVAILABLE_SCHEDULES(v_movie_id);

  LOOP
    FETCH v_ref_cursor INTO v_schedule_id;
    EXIT WHEN v_ref_cursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Available Schedule ID: ' || v_schedule_id);
    SELL_TICKETS(v_schedule_id, v_num_tickets);
  END LOOP;

  CLOSE v_ref_cursor;
END;
```

בסיס הנתונים לפני:

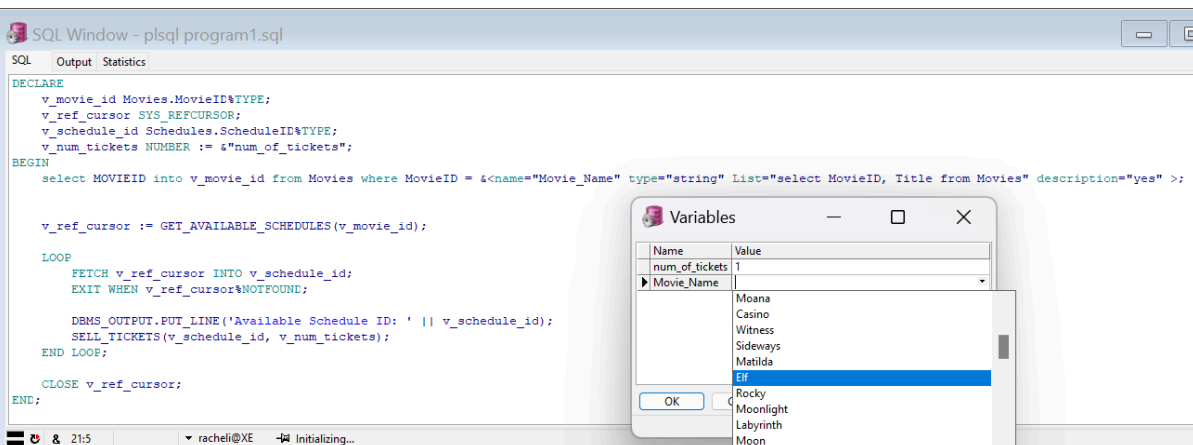
```
SELECT *
FROM TicketSales
WHERE ScheduleID = 317
```

	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
1	105	50.00	N	317	233
2	150	30.00	N	317	465

ניתן לראות שעבור מזהה ההקרנה 317 קיימים 2 כרטיסים שלא נמכרו

הרצת התוכנית:

בחרנו למכור כרטיס 1 מהסרט השייך להקרנה 317



Available Schedule ID: 317
 Tickets where sold

בסיס הנתונים אחרי ההרצה:

נמכר כרטיס אחד מהסרט Elf בהצלחה!

```

SELECT *
FROM TicketSales
WHERE ScheduleID = 317

```

	TICKETID	PRICE	ISSOLD	SCHEDULEID	SUBSCRIBERID
1	105	50.00	Y	317	233
2	150	30.00	N	317	465

2. תיאור מילולי:

התוכנית שלהלן משתמשת בפונקציה Find_Available_Theaters שכתבנו כדי למצוא את חדרי הקולנוע הפנויים בזמן מסויים ומשתמשת בפרוצדורה ADD_SCHEDULE שכתבנו על מנת להוסיף הקרנת סרט נבחר בתאריך ושעה שהמשתמש בוחר.

הקוד:


```

DECLARE
v_movie_id Movies.MovieID%TYPE; -- Replace with actual MovieID
v_date_id Dates.dateID%TYPE; -- Replace with actual DateID
v_theater_cursor SYS_REFCURSOR;

BEGIN
select MOVIEID into v_movie_id from Movies where MovieID = &<name="Movie_Name" type="string" List="select MovieID, Title from Movies" description="yes" >;
select DateID into v_date_id from Dates where DateD = &<name="Date" type="date" List="select DateD from Dates">
and HourH = &<name="Hour" type="string" List="select HourH from Dates">;

-- Call function to get cursor of available theaters
v_theater_cursor := FindAvailableTheaters( v_date_id);

-- Call procedure to add schedule
ADD_SCHEDULE(p_date_id => v_date_id, p_movie_id => v_movie_id, p_theater_cursor => v_theater_cursor);

EXCEPTION
WHEN OTHERS THEN
-- Handle exceptions
DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
IF v_theater_cursor IS NOT NULL THEN
CLOSE v_theater_cursor; -- Close cursor if an error occurs
END IF;
ROLLBACK;
END;

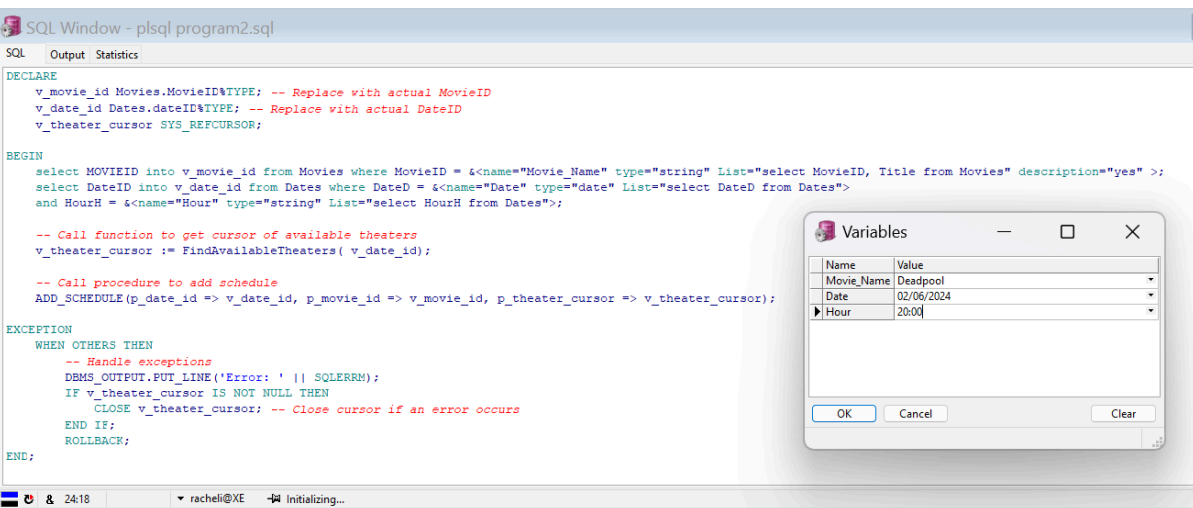
```

בסיס הנתונים לפני:

	SCHEDULEID	THEATERID	AVAILABLESEATS	MOVIEID	DATEID
462	578	124	242	497	256
463	579	109	216	403	380
464	580	102	167	101	452
465	581	123	165	272	157
466	582	113	172	483	125
467	583	113	172	230	572
468	584	104	263	129	122
469	585	105	168	252	579
470	586	116	70	301	105
471	587	123	165	425	339
472	588	130	94	361	368
473	589	125	51	297	193
474	590	107	33	151	553
475	591	118	162	464	289
476	592	102	167	185	295
477	593	118	162	371	250
478	594	112	296	461	242
479	595	124	242	343	503
480	596	109	216	305	303
481	597	129	99	242	271
482	598	128	260	447	430
483	599	121	58	347	175
484	600	121	58	123	456
485	601	121	58	150	150

הרצת התוכנית:

בחרנו סרט, תאריך ושעה מסויימים



New Schedule ID created: 602

בסיס הנתונים אחרי ההרצה:

אפשר לראות את הקרנה 602 שנוספה לנו עם הסרט והזמן שביקשנו!

	SCHEDULEID	THEATERID	AVAILABLESEATS	MOVIEID	DATEID
472	588	130	94	361	368
473	589	125	51	297	193
474	590	107	33	151	553
475	591	118	162	464	289
476	592	102	167	185	295
477	593	118	162	371	250
478	594	112	296	461	242
479	595	124	242	343	503
480	596	109	216	305	303
481	597	129	99	242	271
482	598	128	260	447	430
483	599	121	58	347	175
484	600	121	58	123	456
485	601	121	58	150	150
486	602	121	58	104	106