

מינוי פרויקט
בבסיסי נתונים
סמסטר ב' תשפ"ד

שם המערכת:

תזמורת

שם האגף:

אירועים

מגישות:

chavichaimson@gmail.com
etib2003@gmail.com

חוי בומבר-חיימסון 213419591
אסתר ברנט 325179018

תוכן הענינים

3.....	מערכת: תיאור מילולי
4.....	ERD
5.....	DSD
6.....	createTable
7.....	dropTable
8.....	insertTable
10.....	selectAll
11.....	desc
12.....	מחולל נתונים וייבוא קבצים
16.....	גיבויים ושחזורים
17.....	שאילתות
23.....	שאילתות עם פרמטרים
27.....	אילוצים
29.....	תוכניות, פונקציות ופרוצדורות

תיאור המערכת

שם המערכת:

תזמורת

אגף:

אירועים

תיאור המערכת:

בפרויקט נבנה בסיס נתונים לניהול שירותי תזמורת לאירועים. במערכת ננהל אירוע תוך בחירת מפיק, זמר, כלי תזמורת וכו'.

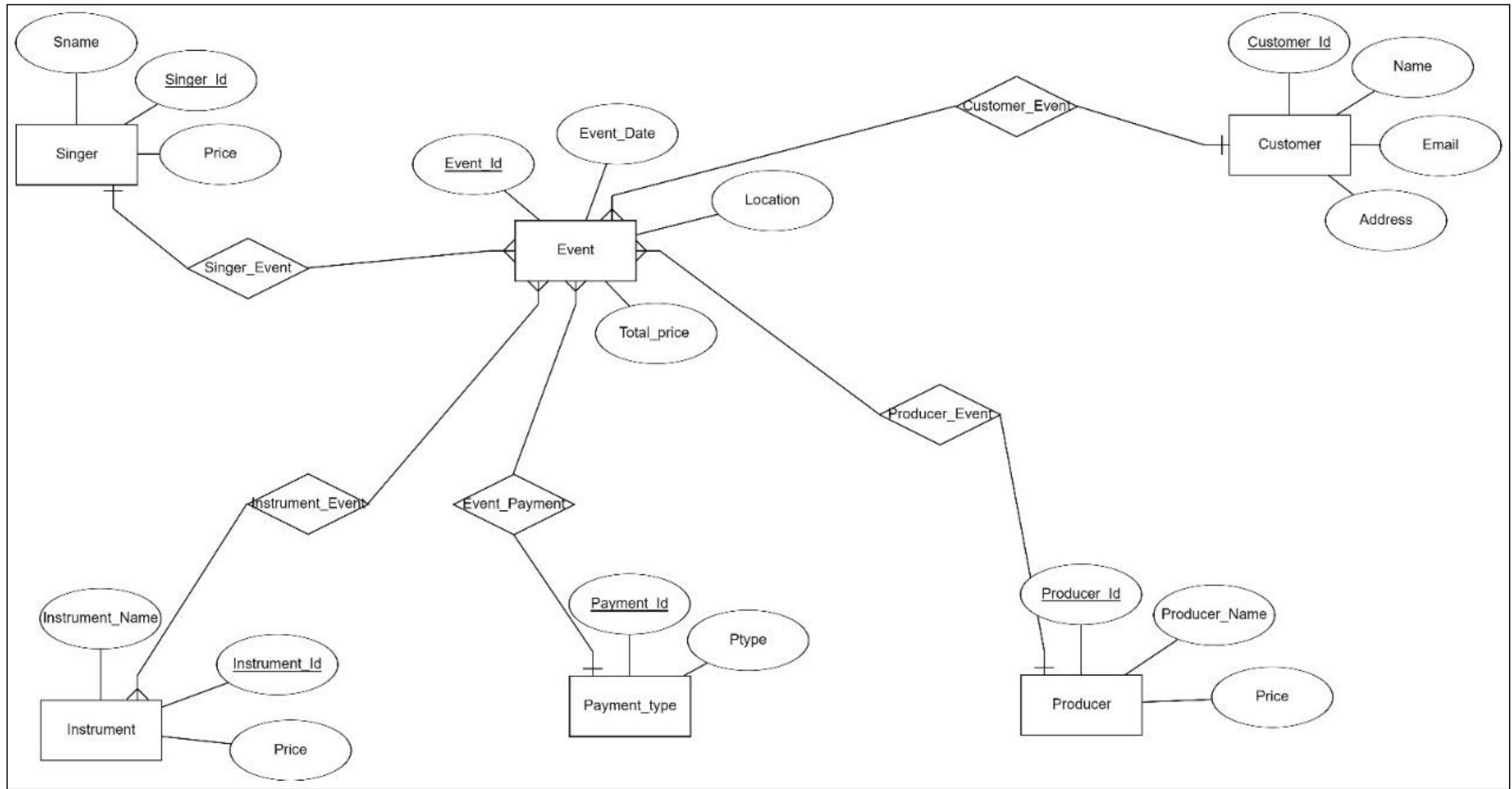
הישויות והשדות:

- אירוע: מספר מזהה, תאריך, מיקום, מחיר, מפיק, זמר, לקוח ואמצעי תשלום.
- מפיק: מספר מזהה, שם ומחיר.
- זמר: מספר מזהה, שם ומחיר.
- כלי תזמורת: מספר מזהה, שם ומחיר.
- לקוח: מספר מזהה, שם, אימייל וכתובת.
- סוג תשלום: מספר מזהה וסוג.

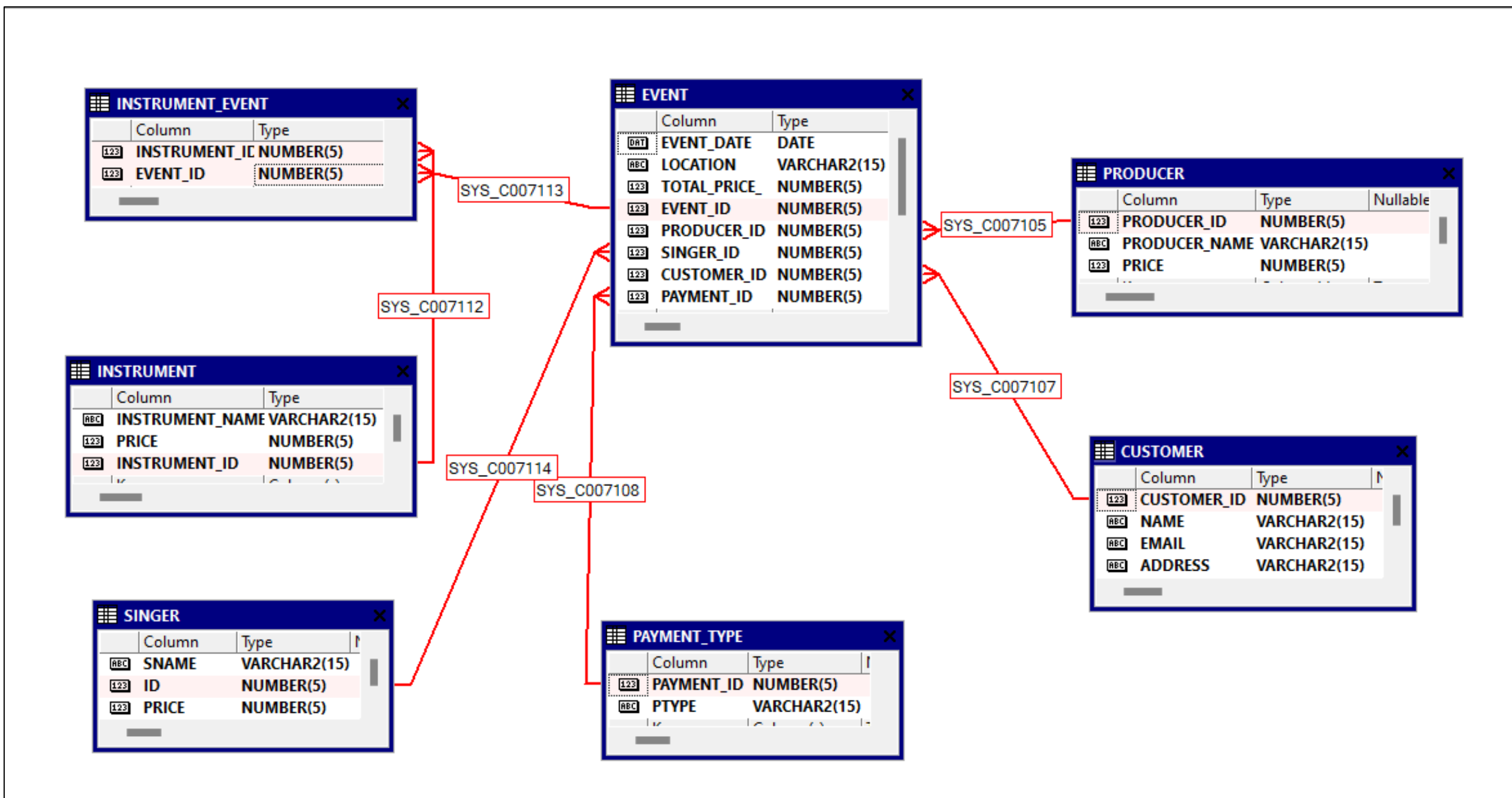
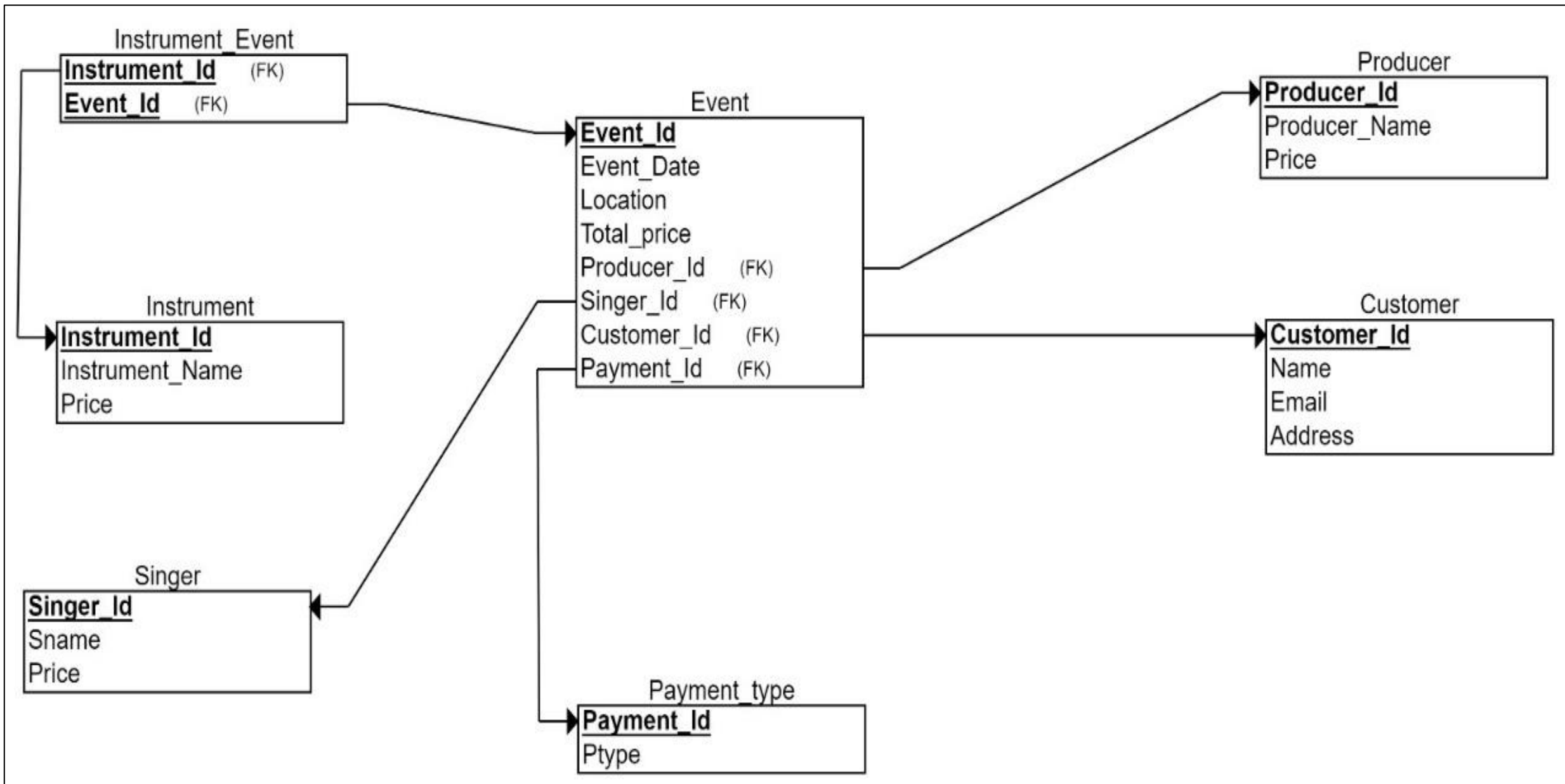
כמו"כ בסיס הנתונים דואג לקשר בין הישויות ע"מ שהאירוע יתנהל כשורה כפי המתוכנן ללא דאגות ובעיות.

דיאגרמות

ERD



DSD



createTables.sql

CREATE TABLE Singer

```
(
  Sname VARCHAR(15) NOT NULL,
  Singer_Id NUMERIC(5) NOT NULL,
  Price NUMERIC(5) NOT NULL,
  PRIMARY KEY (Id)
);
```

CREATE TABLE Customer

```
(
  Customer_Id NUMERIC(5) NOT NULL,
  Name VARCHAR(15) NOT NULL,
  Email VARCHAR(15) NOT NULL,
  Address VARCHAR(15) NOT NULL,
  PRIMARY KEY (Customer_Id)
);
```

CREATE TABLE Instrument

```
(
  Instrument_Name VARCHAR(15) NOT
NULL,
  Price NUMERIC(5) NOT NULL,
  Instrument_Id NUMERIC(5) NOT NULL,
  PRIMARY KEY (Instrument_Id)
);
```

CREATE TABLE Payment_type

```
(
  Payment_Id NUMERIC(5) NOT NULL,
  Ptype VARCHAR(15) NOT NULL,
  PRIMARY KEY (Payment_Id)
);
```

CREATE TABLE Producer

```
(
  Producer_Id NUMERIC(5) NOT NULL,
  Producer_Name VARCHAR(15) NOT
NULL,
  Price NUMERIC(5) NOT NULL,
  PRIMARY KEY (Producer_Id)
);
```

CREATE TABLE Event

```
(
  Event_Date DATE NOT NULL,
  Location VARCHAR(15) NOT NULL,
  Total_price_ NUMERIC(5) NOT NULL,
  Event_Id NUMERIC(5) NOT NULL,
  Producer_Id NUMERIC(5) NOT NULL,
  Singer_Id NUMERIC(5) NOT NULL,
  Customer_Id NUMERIC(5) NOT NULL,
  Payment_Id NUMERIC(5) NOT NULL,
  PRIMARY KEY (Event_Id),
  FOREIGN KEY (Producer_Id)
REFERENCES Producer(Producer_Id),
  FOREIGN KEY (Singer_Id) REFERENCES
Singer(Singer_Id),
  FOREIGN KEY (Customer_Id)
REFERENCES Customer(Customer_Id),
  FOREIGN KEY (Payment_Id)
REFERENCES
Payment_type(Payment_Id)
);
```

CREATE TABLE Instrument_Event

```
(
  Instrument_Id NUMERIC(5) NOT NULL,
  Event_Id NUMERIC(5) NOT NULL,
  PRIMARY KEY (Instrument_Id, Event_Id),
  FOREIGN KEY (Instrument_Id)
REFERENCES Instrument(Instrument_Id),
  FOREIGN KEY (Event_Id) REFERENCES
Event(Event_Id)
);
```

dropTables.sql

```
drop table INSTRUMENT_EVENT;  
drop table EVENT;  
drop table PRODUCER;  
drop table PAYMENT_TYPE;  
drop table INSTRUMENT;  
drop table CUSTOMER;  
drop table SINGER;
```

insertTables.sql - 1

```
insert into singer (id, sname, price)
values (1, ' John Doe', 5000);
```

```
insert into singer (id, sname, price)
values (2, ' Jane Smith', 4500);
```

```
insert into singer (id, sname, price)
values (3, ' Alice Johnson', 4800);
```

```
insert into singer (id, sname, price)
values (4, ' Robert Brown', 5200);
```

```
insert into singer (id, sname, price)
values (5, ' Emily Davis', 4700);
```

```
insert into singer (id, sname, price)
values (6, ' Michael Wilson', 4900);
```

```
insert into singer (id, sname, price)
values (7, ' Sarah Lee', 5300);
```

```
insert into singer (id, sname, price)
values (8, ' David White', 4600);
```

```
insert into singer (id, sname, price)
values (9, ' Laura Harris', 5100);
```

```
insert into singer (id, sname, price)
values (10, ' James Clark', 5400);
```

```
insert into producer (producer_id,
producer_name, price)
values (201, ' Jon Doe', 3000);
```

```
insert into producer (producer_id,
producer_name, price)
values (202, ' Jane Roe', 3200);
```

```
insert into producer (producer_id,
producer_name, price)
values (203, ' Alice Li', 3400);
```

```
insert into producer (producer_id,
producer_name, price)
values (204, ' Bob Lin', 3100);
```

```
insert into producer (producer_id,
producer_name, price)
values (205, ' Carol Yu', 3300);
```

```
insert into producer (producer_id,
producer_name, price)
values (206, ' Dan Kim', 3500);
```

```
insert into producer (producer_id,
producer_name, price)
values (207, ' Eve Wu', 3600);
```

```
insert into producer (producer_id,
producer_name, price)
values (208, ' Frank Ho', 3700);
```

```
insert into producer (producer_id,
producer_name, price)
values (209, ' Grace Ma', 3800);
```

```
insert into producer (producer_id,
producer_name, price)
values (210, ' Henry Xu', 3900);
```


insertTables.sql -2

```
INSERT INTO Payment_type (Payment_Id, Ptype) VALUES (1, 'Cash');
INSERT INTO Payment_type (Payment_Id, Ptype) VALUES (2, 'Credit Card');
INSERT INTO Payment_type (Payment_Id, Ptype) VALUES (3, 'Check');
INSERT INTO Payment_type (Payment_Id, Ptype) VALUES (4, 'PayPal');
INSERT INTO Payment_type (Payment_Id, Ptype) VALUES (5, 'Bank Transfer');
INSERT INTO Payment_type (Payment_Id, Ptype) VALUES (6, 'Bit');
```

```
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (32, 330);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (44, 173);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (24, 344);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (21, 37);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (29, 59);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (9, 90);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (19, 274);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (26, 133);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (15, 133);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (48, 319);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (25, 25);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (27, 107);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (8, 103);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (41, 289);
INSERT INTO Instrument_Event (Instrument_Id, Event_Id) VALUES (2, 50);
```

יש עוד המשך ארוך אבל לא נלאה אתכם. בגיטהאב מצורף הקובץ המלא.

selectAll.sql

```
select * from event;  
select * from instrument_event;  
select * from instrument;  
select * from producer;  
select * from customer;  
select * from singer;  
select * from payment_type;
```

desc

```
Command Window - New
Dialog Editor
Connected as eti@XE

SQL> desc customer
Name          Type          Nullable Default Comments
-----
CUSTOMER_ID   NUMBER(5)
NAME          VARCHAR2(15)
EMAIL         VARCHAR2(15)
ADDRESS       VARCHAR2(15)

SQL> desc producer
Name          Type          Nullable Default Comments
-----
PRODUCER_ID   NUMBER(5)
PRODUCER_NAME VARCHAR2(15)
PRICE         NUMBER(5)

SQL> desc singer
Name          Type          Nullable Default Comments
-----
SNAME         VARCHAR2(15)
SINGER_ID     NUMBER(5)
PRICE         NUMBER(5)

SQL> desc event
Name          Type          Nullable Default Comments
-----
EVENT_DATE    DATE
LOCATION        VARCHAR2(15)
TOTAL_PRICE_  NUMBER(5)
EVENT_ID      NUMBER(5)
PRODUCER_ID   NUMBER(5)
SINGER_ID     NUMBER(5)
CUSTOMER_ID   NUMBER(5)
PAYMENT_ID    NUMBER(5)

SQL> desc instrument
Name          Type          Nullable Default Comments
-----
INSTRUMENT_NAME VARCHAR2(15)
PRICE         NUMBER(5)
INSTRUMENT_ID NUMBER(5)
```

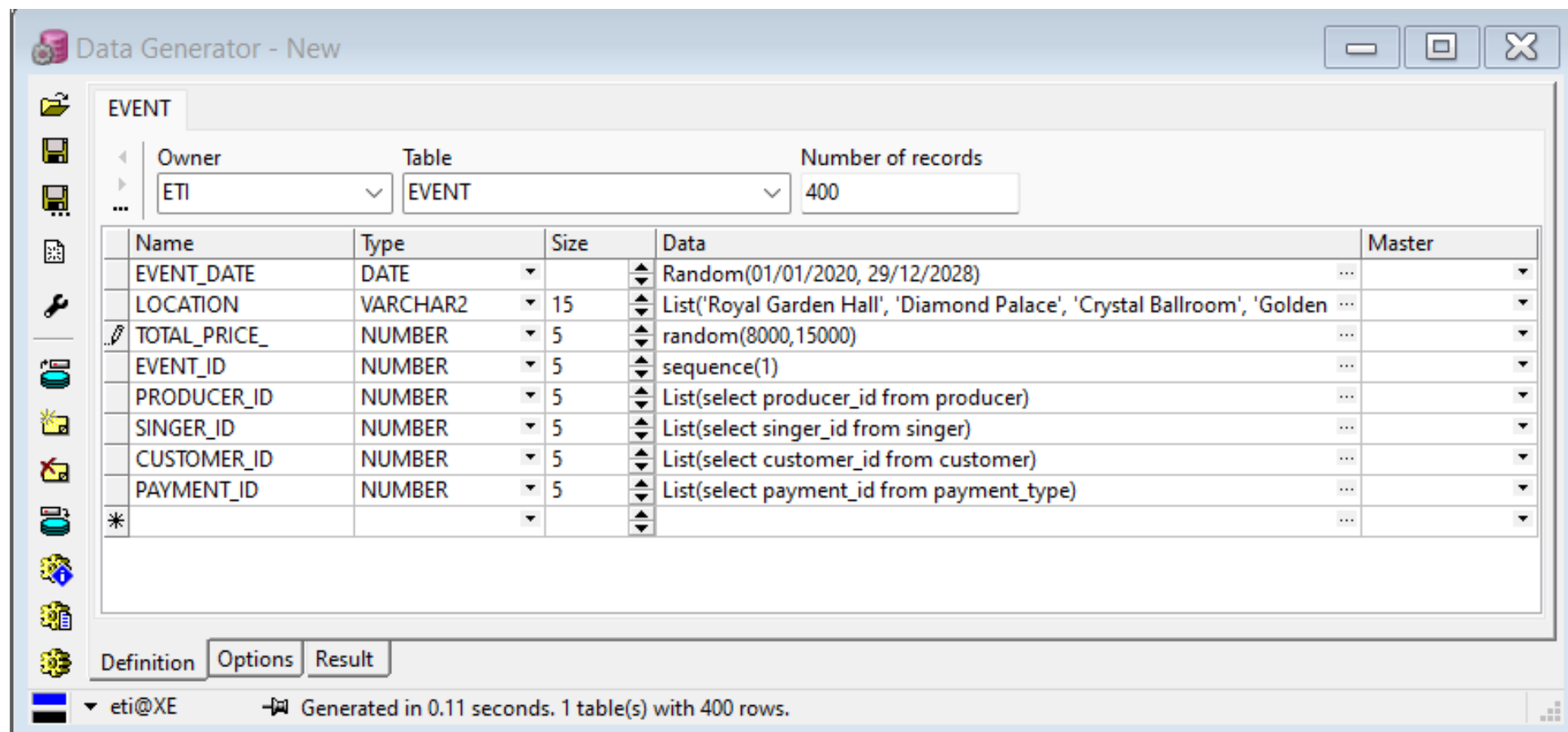
מחולל נתונים וייבוא קבצים

שם המערכת:

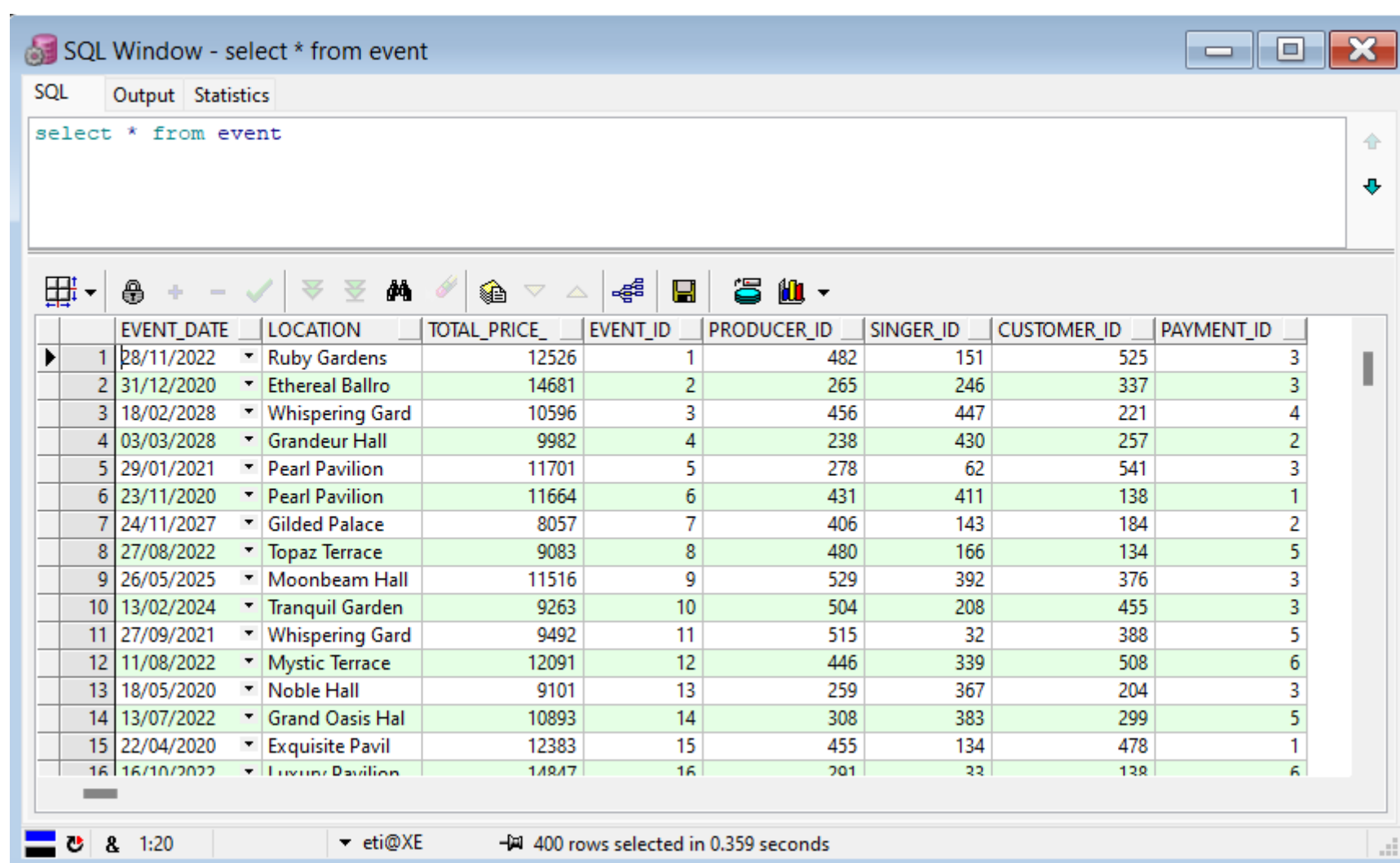
מילאנו את הטבלאות ב3 דרכים: קבצי טקסט, data generator וקוד פייתון.

כעת נציג דוגמא לכל אחת מהדרכים:

דוגמא לשימוש בdata generator עבור הטבלה EVENT:



כעת הטבלה EVENT מכילה 400 רשומות:



דוגמא לשימוש בייבוא נתונים מקובץ טקסט עבור הטבלה CUSTOMER:

101, Alice Brown, alice@ex.com, 123 Maple St
102, Bob Smith, bob@ex.com, 456 Oak St
103, Carol Johnson, carol@ex.com, 789 Pine St
104, David Wilson, david@ex.com, 321 Elm St
105, Eve Davis, eve@ex.com, 654 Cedar St
106, Frank Miller, frank@ex.com, 987 Birch St
107, Grace Lee, grace@ex.com, 111 Cherry St
108, Henry Clark, henry@ex.com, 222 Willow St
109, Irene Lewis, irene@ex.com, 333 Aspen St
110, Jack Walker, jack@ex.com, 444 Poplar St

Text Importer - customerTxt.txt

Data from Textfile Data to Oracle

General

Owner Table ☐ Clear Table

Commit every... 100 ☐ Overwrite duplicates ☒ Ignore duplicates

Initializing Script

Finalizing Script

Fields

Field1 -> CUSTOMER_ID (NUMBER) Field ADDRESS (VARCHAR2)

Field2 -> NAME (VARCHAR2) Fieldtype String

Field3 -> EMAIL (VARCHAR2)

Field4 -> ADDRESS (VARCHAR2)

Create SQL

SQL function

Result Preview

1	2	3	4
101	Alice Brown	alice@ex.com	123 Maple St
102	Bob Smith	bob@ex.com	456 Oak St
103	Carol Johnson	carol@ex.com	789 Pine St
104	David Wilson	david@ex.com	321 Elm St
105	Eve Davis	eve@ex.com	654 Cedar St
106	Frank Miller	frank@ex.com	987 Birch St
107	Grace Lee	grace@ex.com	111 Cherry St

Import Import to Script Close eti@XE customerTxt.txt loaded, 1 KB

כעת הטבלה CUSTOMER מכילה את הרשומות:

SQL Window - select * from customer

SQL Output Statistics

```
select * from customer
```

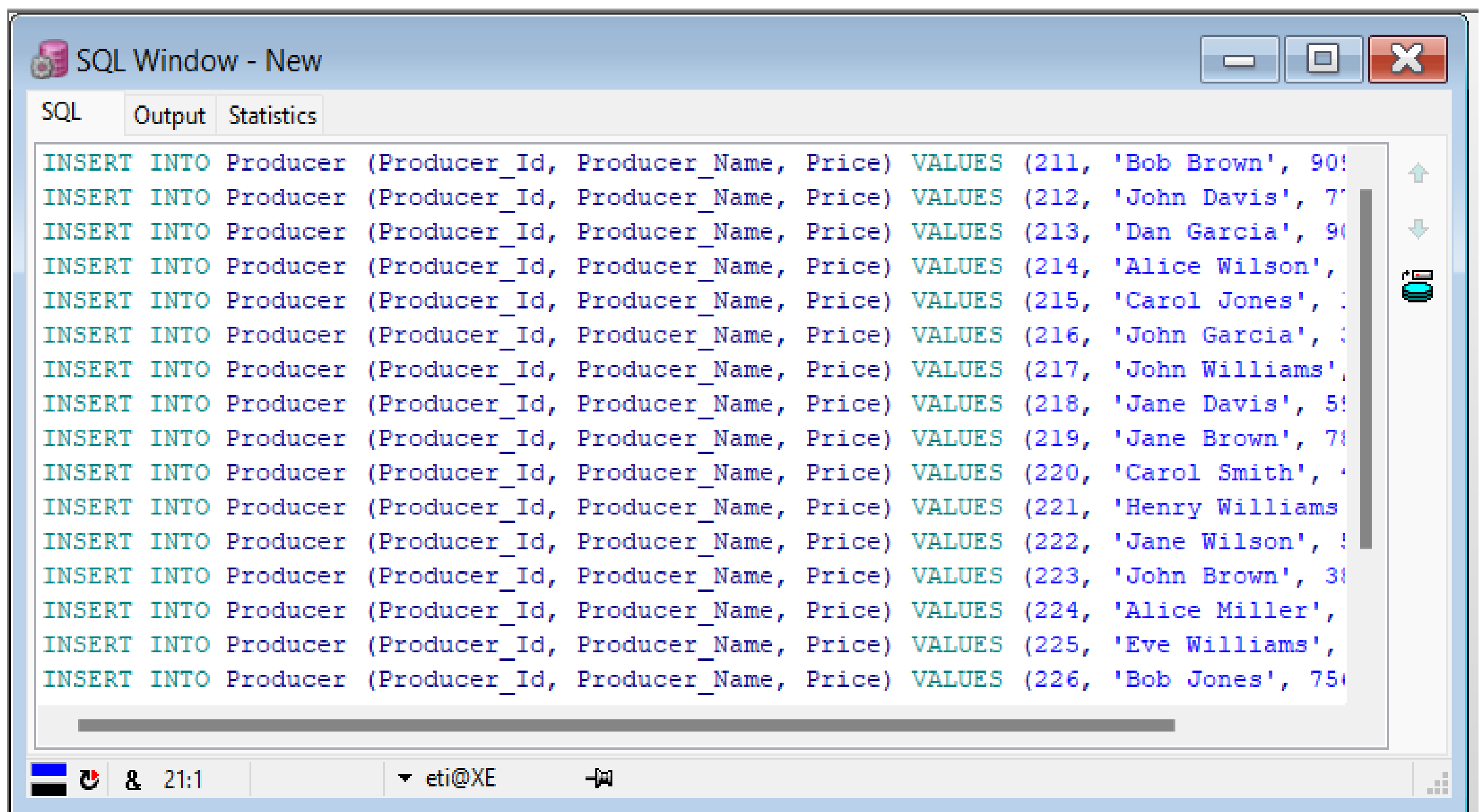
	CUSTOMER_ID	NAME	EMAIL	ADDRESS
1	101	Alice Brown	alice@ex.com	123 Maple St
2	102	Bob Smith	bob@ex.com	456 Oak St
3	103	Carol Johnson	carol@ex.com	789 Pine St
4	104	David Wilson	david@ex.com	321 Elm St
5	105	Eve Davis	eve@ex.com	654 Cedar St
6	106	Frank Miller	frank@ex.com	987 Birch St
7	107	Grace Lee	grace@ex.com	111 Cherry St
8	108	Henry Clark	henry@ex.com	222 Willow St
9	109	Irene Lewis	irene@ex.com	333 Aspen St
10	110	Jack Walker	jack@ex.com	444 Poplar St

1:23 eti@XE 449 rows selected in 0.281 seconds

דוגמא לשימוש בייבוא נתונים מקובץ שנוצר ע"י פייתון עבור הטבלה PRODUCER:

```
1 import random
2
3 # פונקציה ליצירת שמות אקראיים
4 def generate_name():
5     first_names = ['John', 'Jane', 'Alice', 'Bob', 'Carol', 'Dan', 'Eve', 'Frank', 'Grace', 'Henry']
6     last_names = ['Smith', 'Doe', 'Johnson', 'Brown', 'Williams', 'Jones', 'Miller', 'Davis', 'Garcia', 'Wilson']
7     return random.choice(first_names) + ' ' + random.choice(last_names)
8
9 # פונקציה ליצירת מחירים אקראיים
10 def generate_price():
11     return random.randint(1000, 10000) # מחירים בין 1000 ל-10000
12
13 # INSERT יצירת פקודות
14 producers = []
15 for i in range(211, 601):
16     name = generate_name()
17     price = generate_price()
18     insert_statement = f"INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES ({i}, '{name}', {price});"
19     producers.append(insert_statement)
20
21 # כתיבת INSERT פקודות לקובץ
22 with open("producerPy.txt", "w") as file:
23     for insert in producers:
24         file.write(f"{insert}\n")
```

קובץ הטקסט שנוצר:
















The screenshot shows a window titled "SQL Window - New" with three tabs: "SQL", "Output", and "Statistics". The "SQL" tab is active, displaying a list of generated INSERT statements for the "Producer" table. The statements are formatted as follows:

```
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (211, 'Bob Brown', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (212, 'John Davis', 709)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (213, 'Dan Garcia', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (214, 'Alice Wilson', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (215, 'Carol Jones', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (216, 'John Garcia', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (217, 'John Williams', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (218, 'Jane Davis', 509)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (219, 'Jane Brown', 709)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (220, 'Carol Smith', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (221, 'Henry Williams', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (222, 'Jane Wilson', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (223, 'John Brown', 309)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (224, 'Alice Miller', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (225, 'Eve Williams', 909)
INSERT INTO Producer (Producer_Id, Producer_Name, Price) VALUES (226, 'Bob Jones', 759)
```

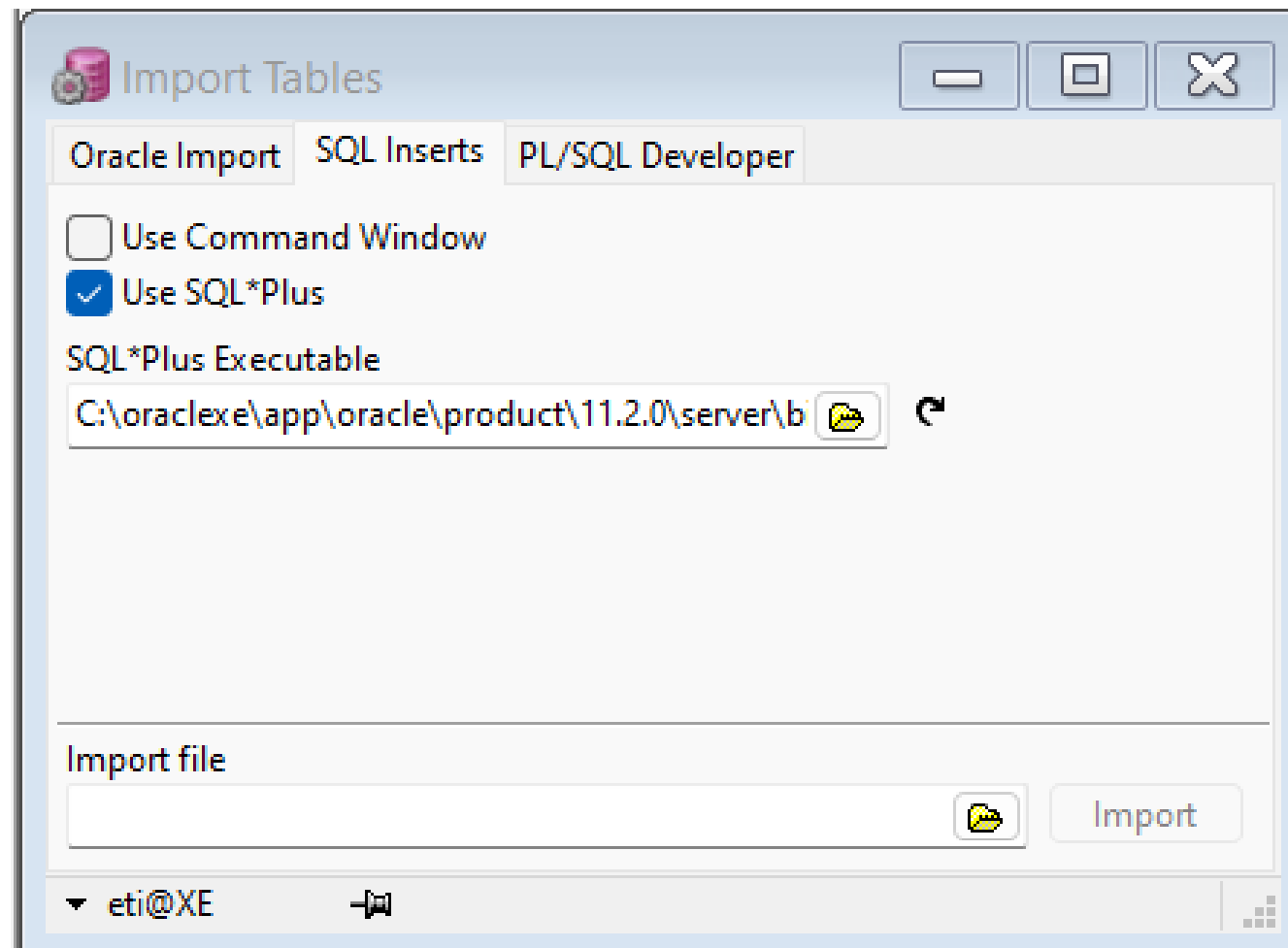
The window also features a status bar at the bottom with icons for undo, redo, and other standard editing functions, along with the text "eti@XE".

כעת הטבלה PRODUCER מכילה את הרשומות:

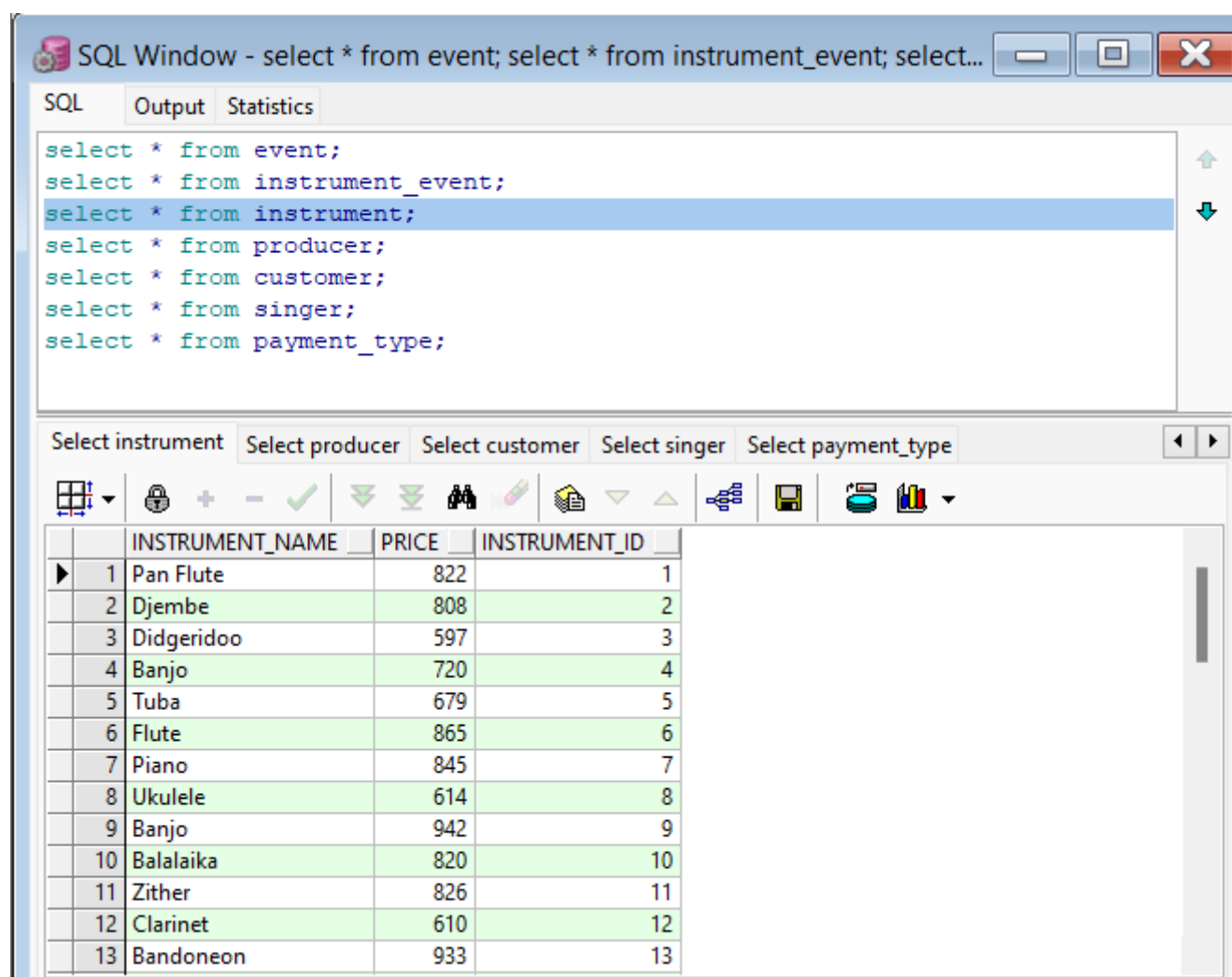
SQL Window - select * from producer				
SQL Output Statistics				
select * from producer				
            				
		PRODUCER_ID	PRODUCER_NAME	PRICE
11		211	Bob Brown	9094
12		212	John Davis	7793
13		213	Dan Garcia	9069
14		214	Alice Wilson	6092
15		215	Carol Jones	1590
16		216	John Garcia	3999
17		217	John Williams	7314
18		218	Jane Davis	5922
19		219	Jane Brown	7800
20		220	Carol Smith	4346
21		221	Henry Williams	9817
22		222	Jane Wilson	5523
23		223	John Brown	3830
24		224	Alice Miller	8357
25		225	Eve Williams	4617
26		226	Bob Jones	7566
27		227	Grace Williams	2408
28		228	Jane Davis	1506
29		229	Bob Johnson	3309

גיבוי ושיחזור

עשינו גיבוי בשיטת sql insert ושמרנו אותו בקובץ backup1.sql
לאחר מכן מחקנו את הטבלאות ע"י drop table ויבאנו מחדש ע"י import tables.



לאחר השחזור בדקנו וראינו שכל הטבלאות והרשומות שבתוכן שוחזרו:



תרגיל 2:

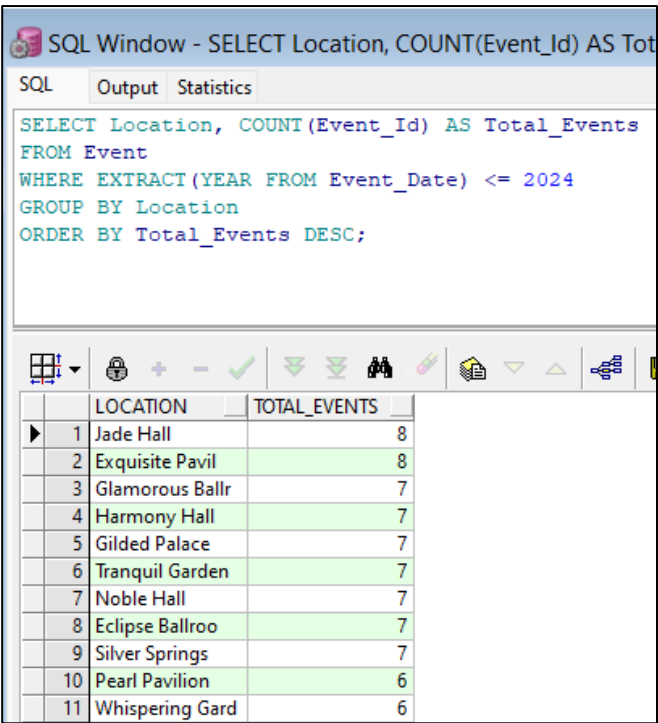
שאלות מורכבות על הטבלאות

שאלות select:

1. הצגת כמות האירועים שהיו בכל אולם עד שנת 2024 (כולל) בסדר יורד:

```
SELECT Location, COUNT(Event_Id) AS Total_Events
FROM Event
WHERE EXTRACT(YEAR FROM Event_Date) <= 2024
GROUP BY Location
ORDER BY Total_Events DESC;
```

תוצאת השאלה:



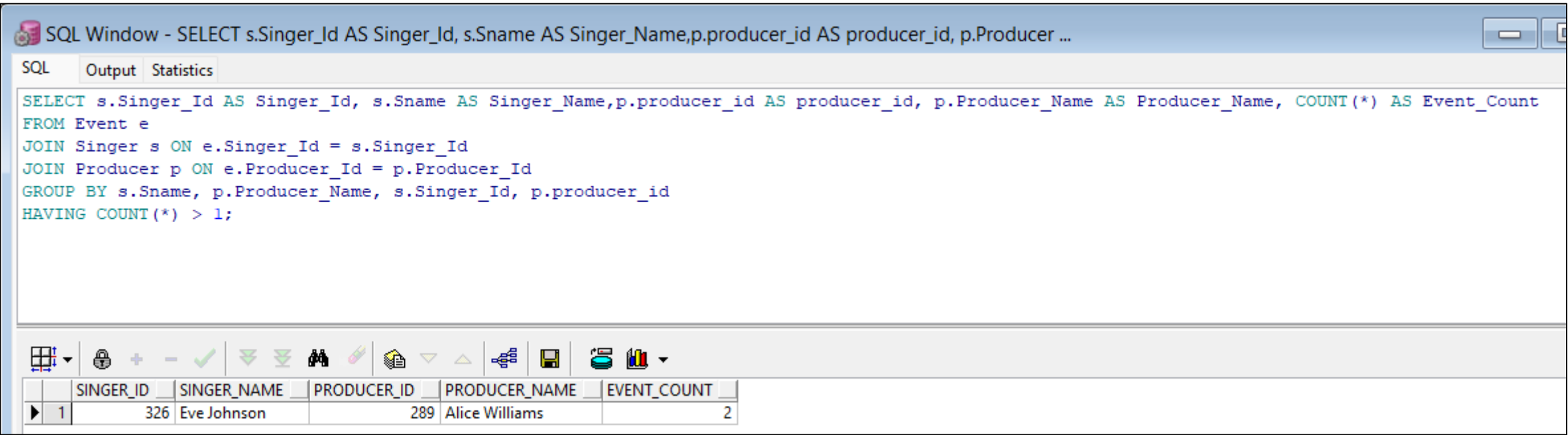
The screenshot shows a SQL window with the following query and output:

	LOCATION	TOTAL_EVENTS
1	Jade Hall	8
2	Exquisite Pavil	8
3	Glamorous Ballr	7
4	Harmony Hall	7
5	Gilded Palace	7
6	Tranquil Garden	7
7	Noble Hall	7
8	Eclipse Ballroo	7
9	Silver Springs	7
10	Pearl Pavilion	6
11	Whispering Gard	6

2. הצגת צמד זמרים ומפיקים שהשתתפו באירוע יחד:

```
SELECT s.Singer_Id AS Singer_Id, s.Sname AS Singer_Name, p.producer_id AS
producer_id, p.Producer_Name AS Producer_Name, COUNT(*) AS Event_Count
FROM Event e
JOIN Singer s ON e.Singer_Id = s.Singer_Id
JOIN Producer p ON e.Producer_Id = p.Producer_Id
GROUP BY s.Sname, p.Producer_Name, s.Singer_Id, p.producer_id
HAVING COUNT(*) > 1;
```

תוצאת השאלה:



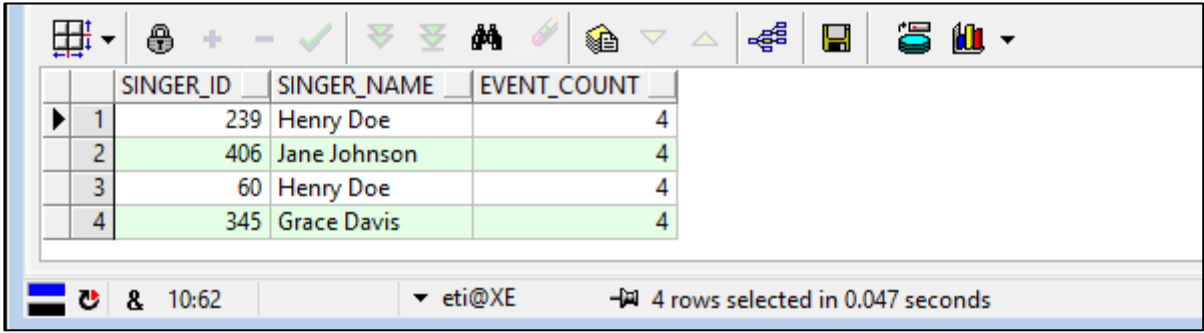
The screenshot shows a SQL window with the following query and output:

	SINGER_ID	SINGER_NAME	PRODUCER_ID	PRODUCER_NAME	EVENT_COUNT
1	326	Eve Johnson	289	Alice Williams	2

3. הצגת הזמרים הכי פופולריים לפי כמות האירועים בהם הופיעו:

```
WITH MaxEvents AS (  
    SELECT s.Singer_Id, s.Sname AS Singer_Name, COUNT(*) AS Event_Count  
    FROM Event e  
    JOIN Singer s ON e.Singer_Id = s.Singer_Id  
    GROUP BY s.Singer_Id, s.Sname  
)  
  
SELECT *  
FROM MaxEvents  
WHERE Event_Count = (SELECT MAX(Event_Count) FROM MaxEvents);
```

תוצאת השאילתא:



	SINGER_ID	SINGER_NAME	EVENT_COUNT
1	239	Henry Doe	4
2	406	Jane Johnson	4
3	60	Henry Doe	4
4	345	Grace Davis	4

4. הצגת ממוצע עלות אירועים בקיץ, חורף ובשאר העונות:

```
WITH Seasons AS (  
    SELECT  
        CASE  
            WHEN TO_CHAR(Event_Date, 'MM') IN ('5','06','07','08','9') THEN  
                'Summer'  
            WHEN TO_CHAR(Event_Date, 'MM') IN ('11','12','01','02') THEN 'Winter'  
            ELSE 'Other'  
        END AS Season,  
        Total_Price_  
    FROM  
        Event  
)  
SELECT  
    Season,  
    AVG(Total_Price_) AS Average_Price  
FROM  
    Seasons  
GROUP BY  
    Season  
ORDER BY  
    Average_Price DESC;
```

תוצאת השאילתא:

	SEASON	AVERAGE_PRICE
1	Summer	12210.265060241
2	Winter	11698.0536912752
3	Other	11465.5297619048

שאלות update:

1. עדכון שדה המחיר הכולל של אירוע (סכימת מחירי מפיק, זמר וכלי הנגינה שבאירוע):

```
UPDATE Event e
SET e.Total_price_ = (
    SELECT p.Price + s.Price + (
        SELECT COALESCE(SUM(i.Price), 0)
        FROM Instrument_Event ie, Instrument i
        WHERE ie.Event_Id = e.Event_Id AND
ie.Instrument_Id = i.Instrument_Id)
    FROM Producer p, Singer s
    WHERE e.Producer_Id = p.Producer_Id AND e.Singer_Id = s.Singer_Id
);
```

תוצאת השאילתא:

SQL Window - UPDATE Event e SET e.Total_price_ = (SELECT p.Price + s.Price + (SELECT COALESCE(SUM(i.Price), ...

SQL Output Statistics

UPDATE Event e
SET e.Total_price_ = (
 SELECT p.Price + s.Price + (
 SELECT COALESCE(SUM(i.Price), 0)
 FROM Instrument_Event ie, Instrument i
 WHERE ie.Event_Id = e.Event_Id AND ie.Instrument_Id = i.Instrument_Id)
 FROM Producer p, Singer s
 WHERE e.Producer_Id = p.Producer_Id AND e.Singer_Id = s.Singer_Id
);

select * from event;
select * from producer;
select * from singer;
select * from instrument_event;
select * from instrument;

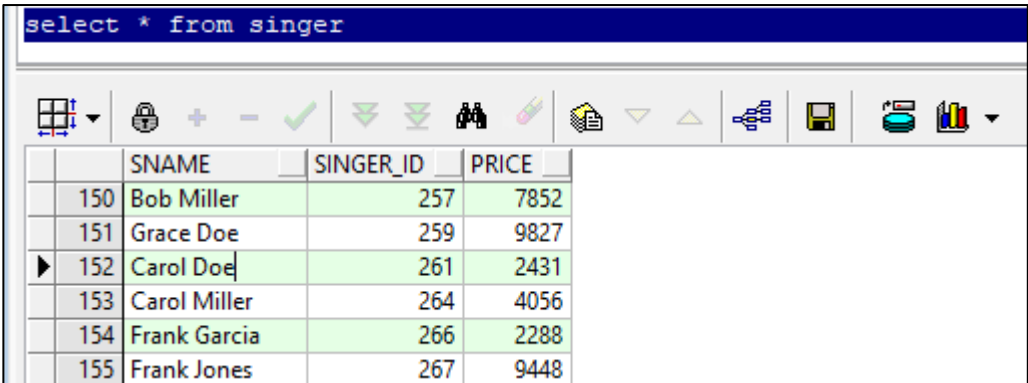
Select event Select producer Select singer Select instrument_event Select instrument

	EVENT_DATE	LOCATION	TOTAL_PRICE_	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
1	28/11/2022	Ruby Gardens	11555	1	482	151	525	3
2	31/12/2020	Ethereal Ballro	9969	2	265	246	337	3
3	18/02/2028	Whispering Gard	9851	3	456	447	221	4
4	03/03/2028	Grandeur Hall	15002	4	238	430	257	2
5	29/01/2021	Pearl Pavilion	11647	5	278	62	541	3
6	23/11/2020	Pearl Pavilion	14261	6	431	411	138	1
7	24/11/2027	Gilded Palace	14828	7	406	143	184	2
8	27/08/2022	Topaz Terrace	14103	8	480	166	134	5
9	26/05/2025	Moonbeam Hall	16433	9	529	392	376	3
10	13/02/2024	Tranquil Garden	16894	10	504	208	455	3
11	27/09/2021	Whispering Gard	7867	11	515	32	388	5
12	11/08/2022	Mystic Terrace	11666	12	446	339	508	6
13	18/05/2020	Noble Hall	17883	13	259	367	204	3
14	13/07/2022	Grand Oasis Hal	18277	14	308	383	299	5
15	22/04/2020	Exquisite Pavil	17404	15	455	134	478	1
16	16/10/2022	Luxury Pavilion	13309	16	291	33	138	6
17	19/04/2027	Silver Springs	9017	17	420	303	408	6
18	03/11/2025	Noble Hall	13185	18	200	226	411	5

2. הכפלה פי 1.1 של מחירי הזמרים (העלאת המחיר) שהופיעו יותר מפעמיים ב-4 שנים האחרונות:

```
UPDATE singer
SET price=price*1.1
WHERE singer_id IN(
  SELECT singer_id FROM event
  WHERE (EXTRACT(YEAR FROM Event_Date) BETWEEN 2020 AND 2024)
  GROUP BY singer_id
  HAVING COUNT(*) > 2);
```

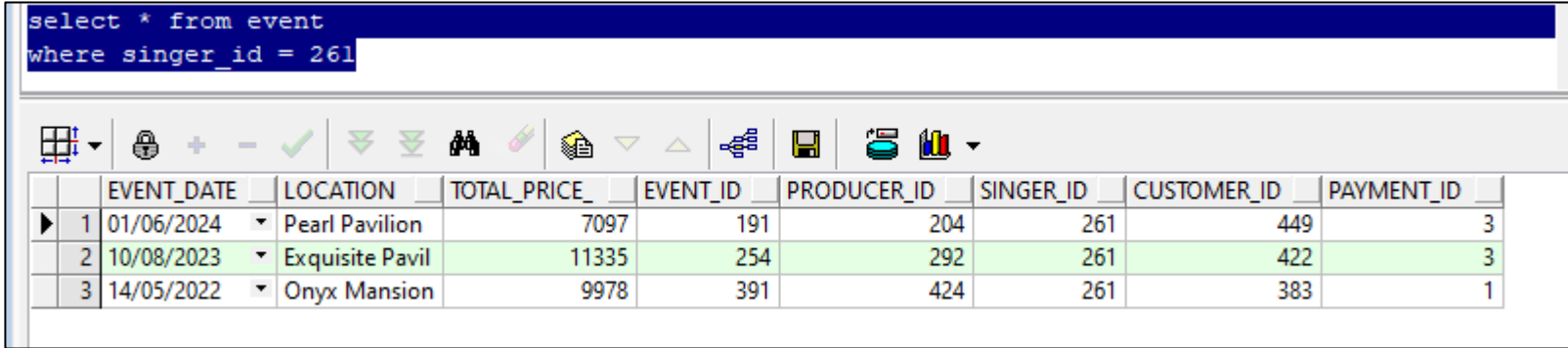
מחיר זמר 261 לפני הרצת השאילתא:



The screenshot shows a database query window with the command 'select * from singer'. The result table has columns SNAME, SINGER_ID, and PRICE. The data is as follows:

	SNAME	SINGER_ID	PRICE
150	Bob Miller	257	7852
151	Grace Doe	259	9827
152	Carol Doe	261	2431
153	Carol Miller	264	4056
154	Frank Garcia	266	2288
155	Frank Jones	267	9448

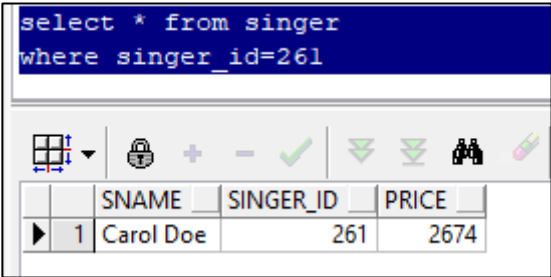
רואים שהזמר עונה על הקריטריון:



The screenshot shows a database query window with the command 'select * from event where singer_id = 261'. The result table has columns EVENT_DATE, LOCATION, TOTAL_PRICE, EVENT_ID, PRODUCER_ID, SINGER_ID, CUSTOMER_ID, and PAYMENT_ID. The data is as follows:

	EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
1	01/06/2024	Pearl Pavilion	7097	191	204	261	449	3
2	10/08/2023	Exquisite Pavil	11335	254	292	261	422	3
3	14/05/2022	Onyx Mansion	9978	391	424	261	383	1

לאחר הרצת השאילתא:



The screenshot shows a database query window with the command 'select * from singer where singer_id=261'. The result table has columns SNAME, SINGER_ID, and PRICE. The data is as follows:

	SNAME	SINGER_ID	PRICE
1	Carol Doe	261	2674

שאלות delete:

1. מחיקת כל הזמרים שלא הופיעו באירועים החל משנת 2021 (מפאת חוסר רלוונטיות):

```
DELETE FROM Singer
WHERE Singer_Id NOT IN (
    SELECT DISTINCT e.Singer_Id
    FROM Event e
    WHERE e.Event_Date >= DATE '2021-01-01'
);
```

לדוגמא: זמר 15 הופיע פעם אחרונה בשנת 2020, ולכן צריך להמחק:

select * from event

	EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
278	15/01/2020	Jade Hall	8906	278	476	388	518	1
355	24/01/2020	Radiant Hall	9975	355	425	115	219	1
342	27/01/2020	Emerald Manor	11698	342	464	393	545	1
392	01/03/2020	Regal Ballroom	4394	392	595	334	198	5
141	07/03/2020	Breathtaking Te	8928	141	542	142	101	4
345	16/04/2020	Elegant Terrace	4172	345	502	441	279	3
15	22/04/2020	Exquisite Pavil	17404	15	455	134	478	1
117	04/05/2020	Jade Hall	11465	117	593	449	399	3
13	18/05/2020	Noble Hall	17883	13	259	367	204	3
183	15/06/2020	Celestial Terra	5989	183	528	284	192	2
218	24/06/2020	Silver Springs	18191	218	594	155	243	4
349	29/07/2020	Sapphire Hall	11091	349	210	135	213	6
308	29/07/2020	Amethyst Ballro	18648	308	325	20	486	3
222	05/08/2020	Luxury Pavilion	10311	222	536	15	429	5
398	07/08/2020	Emerald Manor	15986	398	428	95	265	2

לאחר הרצת השאילתא רואים שזמר 15 כבר לא מופיע בטבלת האירועים ובטבלת הזמרים:

select * from event where singer_id=15;

EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
------------	----------	-------------	----------	-------------	-----------	-------------	------------

select * from singer;

	SNAME	SINGER_ID	PRICE
1	Alice Johnson	3	4800
2	Michael Wilson	6	4900
3	David White	8	4600
4	Laura Harris	9	5100
5	James Clark	10	5400
6	Frank Garcia	11	4911
7	Dan Smith	13	6473
8	Dan Smith	21	9381
9	Jane Williams	23	5408
10	Carol Williams	25	4508

2. מחיקת כל האירועים שאמורים להתקיים באולם "Whispering Gard" בין החודשים 4 ל 7 בשנת 2025 עקב סגירה זמנית של האולם לצורך שיפוצים:

```
DELETE FROM event
WHERE location = 'Whispering Gard' AND (EXTRACT(YEAR FROM Event_Date) = 2025 AND
EXTRACT(MONTH FROM Event_Date) BETWEEN 4 AND 7);
```

```
select * from event
WHERE location = 'Whispering Gard';
```

טבלת האירועים לפני הרצת השאילתא:

SQL Window - select * FROM event WHERE location = 'Whispering Gard';								
SQL Output Statistics								
<pre>select * FROM event WHERE location = 'Whispering Gard';</pre>								
	EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
2	27/09/2021	Whispering Gard	7867	11	515	32	388	5
7	02/05/2022	Whispering Gard	13865	157	292	195	243	4
3	14/06/2024	Whispering Gard	14566	23	328	302	214	5
4	03/10/2024	Whispering Gard	17079	58	359	139	424	4
10	08/12/2024	Whispering Gard	14923	303	286	224	388	2
12	09/12/2024	Whispering Gard	14307	320	363	247	538	1
5	29/03/2025	Whispering Gard	13475	71	433	211	259	5
11	27/06/2025	Whispering Gard	9185	318	250	287	362	6
13	07/11/2025	Whispering Gard	16140	370	562	226	106	6
6	20/01/2027	Whispering Gard	17781	104	423	309	324	3
9	13/10/2027	Whispering Gard	8490	285	498	111	109	3
1	18/02/2028	Whispering Gard	9851	3	456	447	221	4
8	07/04/2028	Whispering Gard	11775	202	322	371	509	2

לאחר הרצת השאילתא רואים שהאירועים שהיו בתאריכים הנ"ל נמחקו:

select * from event WHERE location = 'Whispering Gard';								
	EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
2	27/09/2021	Whispering Gard	7867	11	515	32	388	5
7	02/05/2022	Whispering Gard	13865	157	292	195	243	4
3	14/06/2024	Whispering Gard	14566	23	328	302	214	5
4	03/10/2024	Whispering Gard	17079	58	359	139	424	4
10	08/12/2024	Whispering Gard	14923	303	286	224	388	2
11	09/12/2024	Whispering Gard	14307	320	363	247	538	1
5	29/03/2025	Whispering Gard	13475	71	433	211	259	5
12	07/11/2025	Whispering Gard	16140	370	562	226	106	6
6	20/01/2027	Whispering Gard	17781	104	423	309	324	3
9	13/10/2027	Whispering Gard	8490	285	498	111	109	3
1	18/02/2028	Whispering Gard	9851	3	456	447	221	4
8	07/04/2028	Whispering Gard	11775	202	322	371	509	2

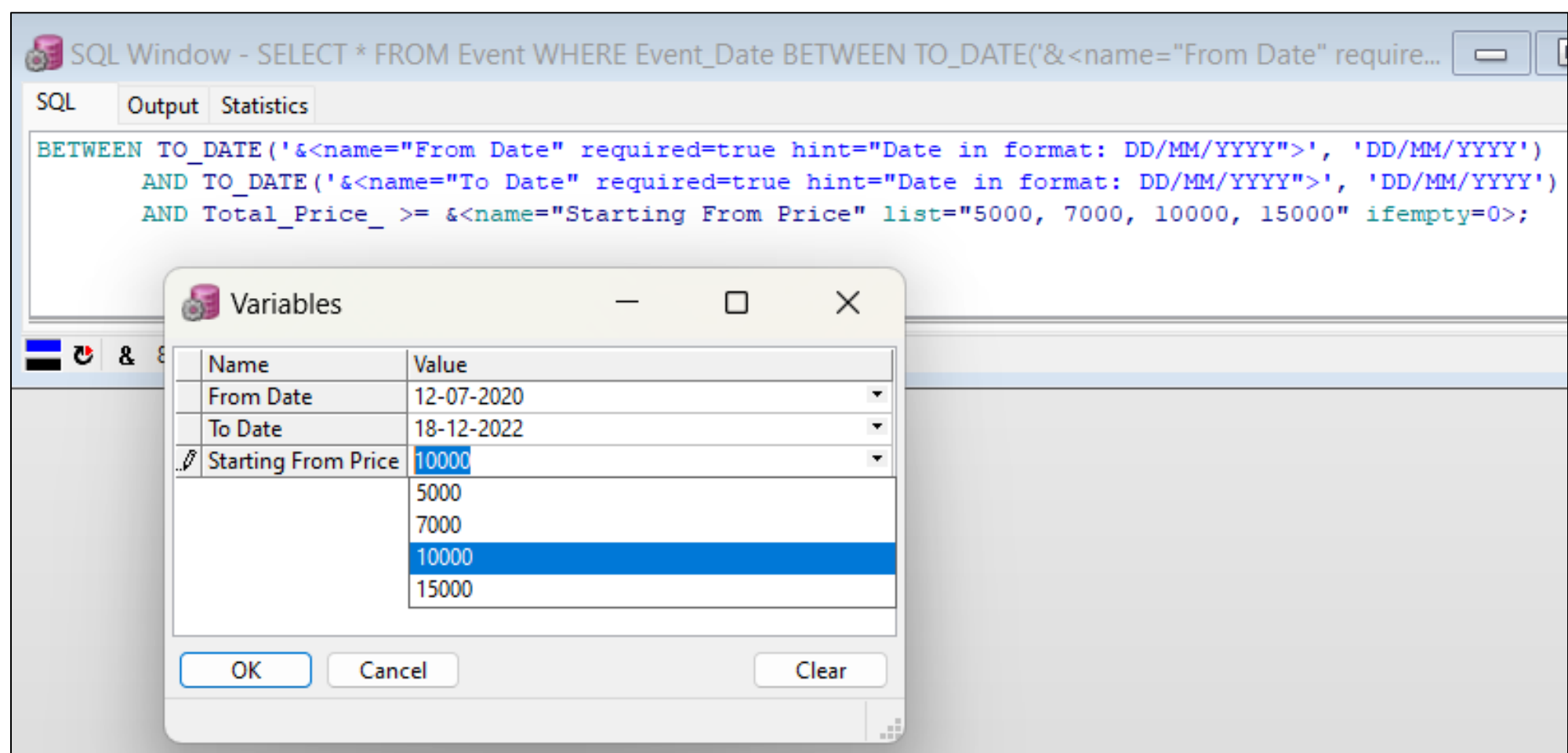
שאלות עם פרמטרים

1. שאלת select:

בחירת כל האירועים שהם בין התאריכים שמכניס המשתמש ומחירתם גדול מהמחיר שביקש המשתמש:

```
SELECT *
FROM Event
WHERE Event_Date
BETWEEN TO_DATE('&<name="From Date" required=true hint="Date in format:
DD/MM/YYYY">', 'DD/MM/YYYY')
AND TO_DATE('&<name="To Date" required=true hint="Date in format:
DD/MM/YYYY">', 'DD/MM/YYYY')
AND Total_Price_ >= &<name="Starting From Price" list="5000, 7000, 10000,
15000" ifempty=0>;
```

הרצת השאלת:



תוצאת השאלת:

SQL Window - SELECT * FROM Event WHERE Event_Date BETWEEN TO_DATE('&<name="From Date" require...

SQLOutputStatistics

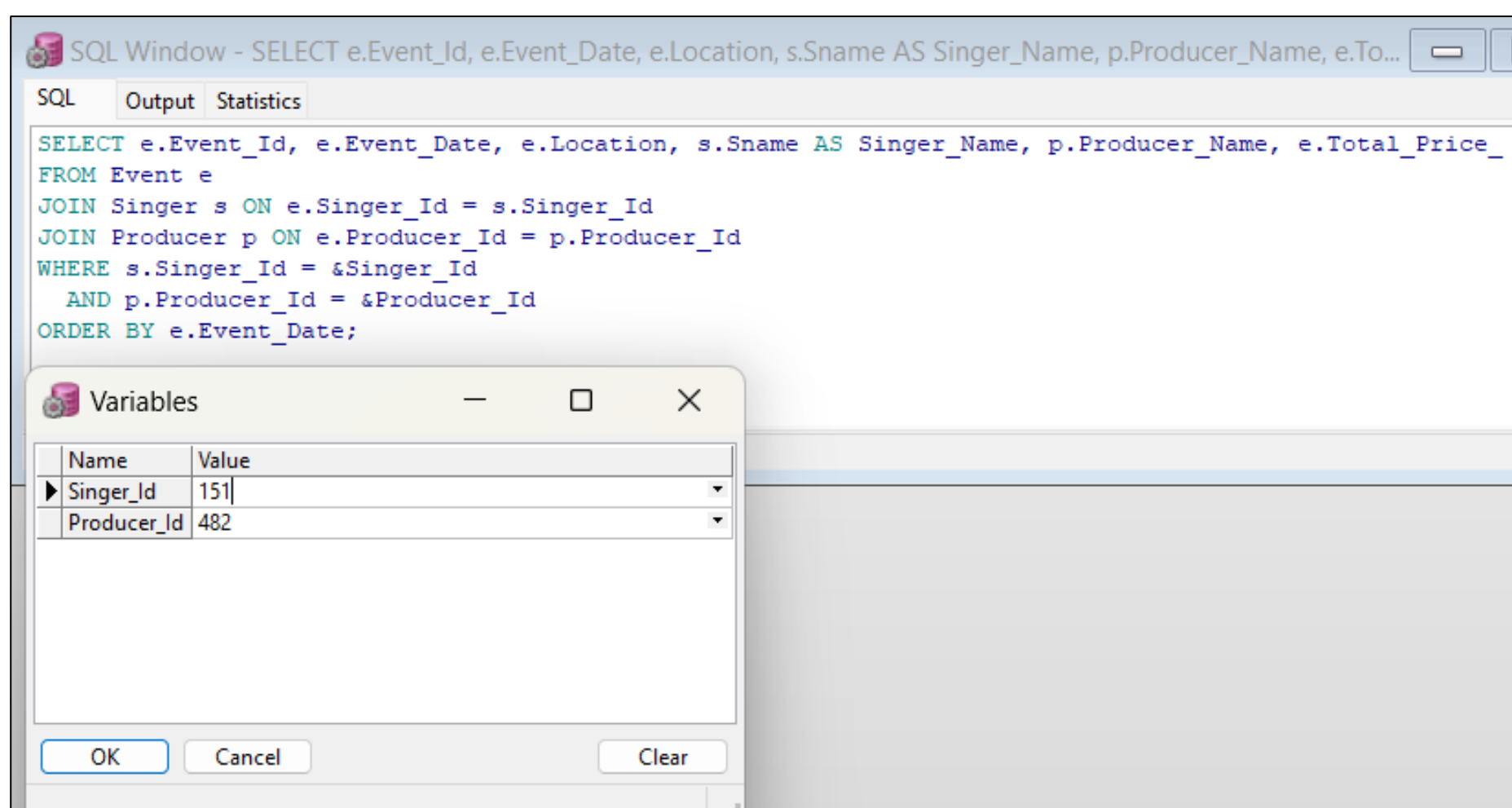
BETWEEN TO_DATE('&<name="From Date" required=true hint="Date in format: DD/MM/YYYY">', 'DD/MM/YYYY')
AND TO_DATE('&<name="To Date" required=true hint="Date in format: DD/MM/YYYY">', 'DD/MM/YYYY')
AND Total_Price_ >= &<name="Starting From Price" list="5000, 7000, 10000, 15000" ifempty=0>;

2. שאילתת select:

בחירת כל האירועים בהם השתתפו יחד מפיק וזמר שהוכנסו כפרמטרים על ידי המשתמש:

```
SELECT e.Event_Id, e.Event_Date, e.Location, s.Sname AS Singer_Name,  
p.Producer_Name, e.Total_Price_  
FROM Event e  
JOIN Singer s ON e.Singer_Id = s.Singer_Id  
JOIN Producer p ON e.Producer_Id = p.Producer_Id  
WHERE s.Singer_Id = &Singer_Id  
AND p.Producer_Id = &Producer_Id  
ORDER BY e.Event_Date;
```

הרצת השאילתא:



תוצאת השאילתא:

The screenshot shows the SQL Window with the same query as above. The results are displayed in a table below the query editor.

	EVENT_ID	EVENT_DATE	LOCATION	SINGER_NAME	PRODUCER_NAME	TOTAL_PRICE
1	1	28/11/2022	Ruby Gardens	Dan Davis	Bob Johnson	11555

At the bottom of the window, a status bar indicates: 1 row selected in 0.031 seconds.

3. שאילתת update:

מפיק מסוים הפסיק לעבוד, ולכן נצטרך להחליף את המפיק בכל האירועים שבהם הוא היה אמור לקחת חלק:

```
UPDATE Event
SET Producer_Id = (
    SELECT Producer_Id
    FROM (
        SELECT Producer_Id
        FROM Producer
        WHERE Producer_Id != &producer_id
        ORDER BY DBMS_RANDOM.RANDOM
    )
WHERE ROWNUM = 1
)
```

לפני ההרצה:

	EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
191	02/12/2028	Topaz Terrace	7447	356	412	238	351	5
143	02/03/2026	Diamond Palace	7165	276	412	292	218	1
112	07/03/2028	Imperial Mansio	6411	224	420	335	316	5
5	19/04/2027	Silver Springs	9017	17	420	303	408	6
85	19/07/2025	Regal Ballroom	6948	173	420	37	376	3
36	01/08/2024	Elegant Terrace	16039	72	423	429	210	5
57	20/01/2027	Whispering Gard	17781	104	423	309	324	3

```
UPDATE Event
SET Producer_Id = (
    SELECT Producer_Id
    FROM (
        SELECT Producer_Id
        FROM Producer
        WHERE Producer_Id != &producer_id
        ORDER BY DBMS_RANDOM.RANDOM
    )
WHERE ROWNUM = 1
)
WHERE Event_Date > SYSDATE AND Producer_Id = &producer_id;
```

Name	Value
producer_id	420

הרצת השאילתא:

תוצאת השאילתא – רואים שהמפיק הוחלף:

```
select * from event
where event_id = 224 or event_id = 17 or event_id = 173
```

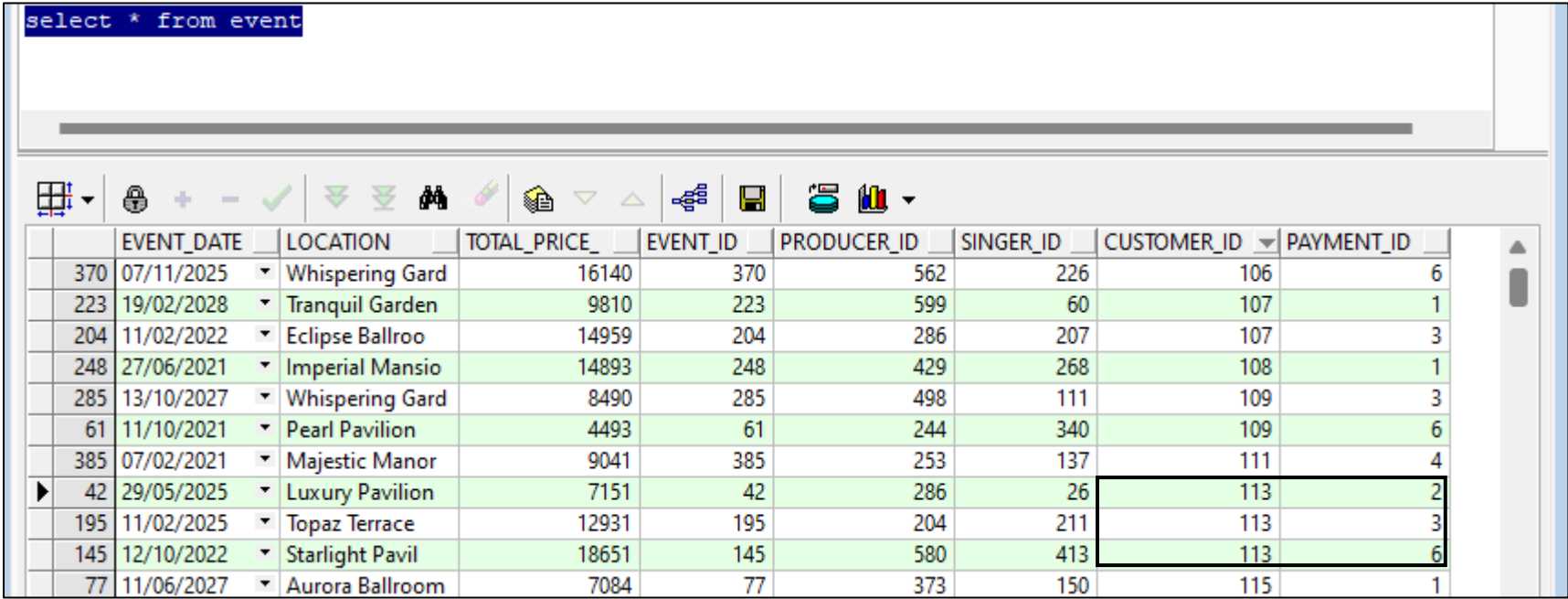
	EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
1	19/04/2027	Silver Springs	9017	17	326	303	408	6
2	19/07/2025	Regal Ballroom	6948	173	326	37	376	3
3	07/03/2028	Imperial Mansio	6411	224	326	335	316	5

4. שאילתת update:

לקוח רוצה לשנות את אמצעי התשלום שלו, ולכן עלינו לעדכן זאת בטבלת האירועים:

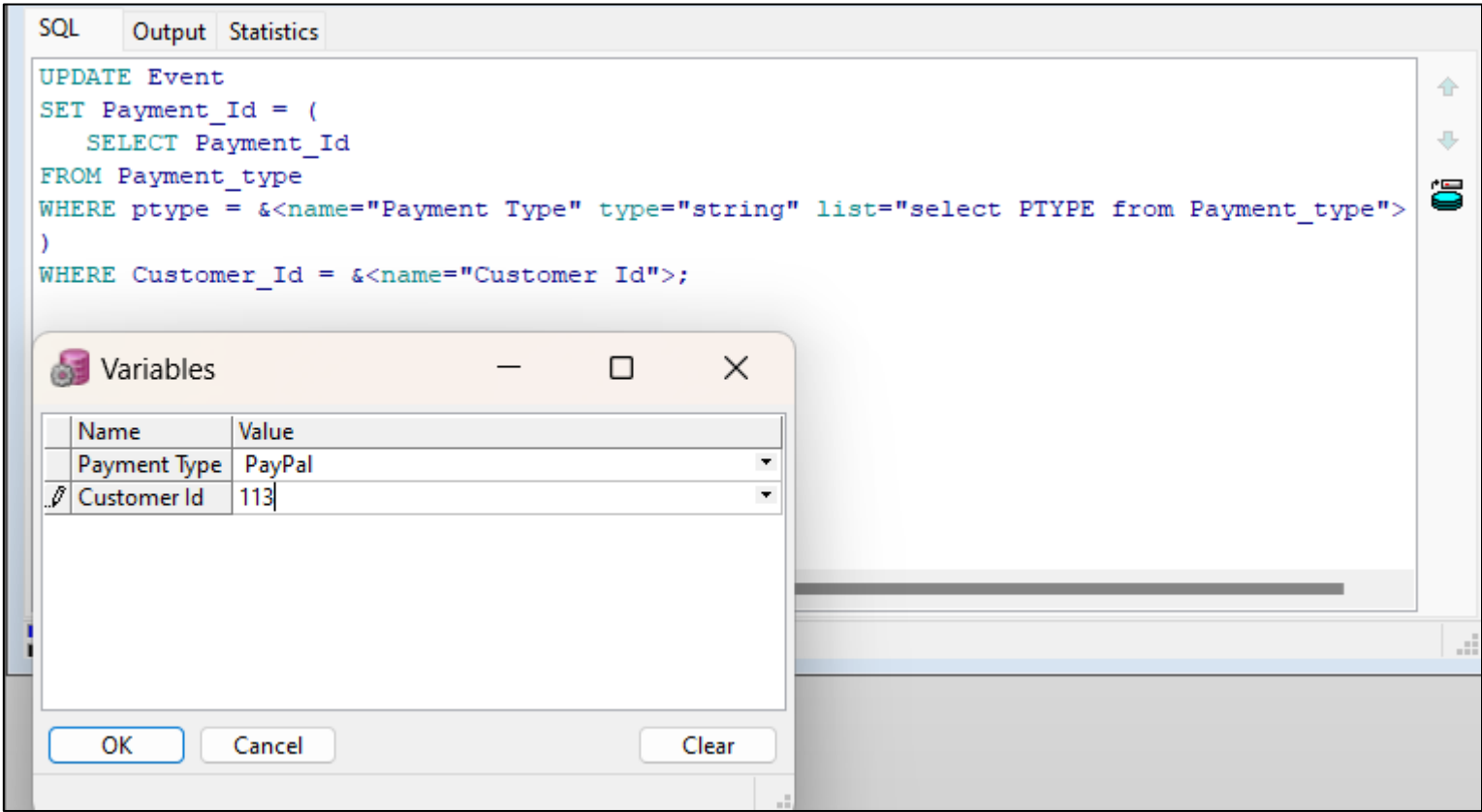
```
UPDATE Event
SET Payment_Id = (
    SELECT Payment_Id
FROM Payment_type
WHERE ptype = &<name="Payment Type" type="string" list="select PTYPE from Payment_type">
)
WHERE Customer_Id = &<name="Customer Id">;
```

לפני ההרצה:



	EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
370	07/11/2025	Whispering Gard	16140	370	562	226	106	6
223	19/02/2028	Tranquil Garden	9810	223	599	60	107	1
204	11/02/2022	Eclipse Ballroo	14959	204	286	207	107	3
248	27/06/2021	Imperial Mansio	14893	248	429	268	108	1
285	13/10/2027	Whispering Gard	8490	285	498	111	109	3
61	11/10/2021	Pearl Pavilion	4493	61	244	340	109	6
385	07/02/2021	Majestic Manor	9041	385	253	137	111	4
42	29/05/2025	Luxury Pavilion	7151	42	286	26	113	2
195	11/02/2025	Topaz Terrace	12931	195	204	211	113	3
145	12/10/2022	Starlight Pavil	18651	145	580	413	113	6
77	11/06/2027	Aurora Ballroom	7084	77	373	150	115	1

הרצת השאילתא:



SQL Output Statistics

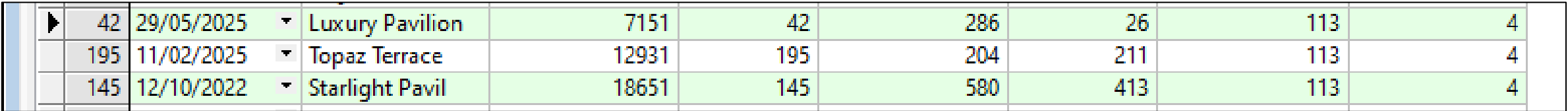
```
UPDATE Event
SET Payment_Id = (
    SELECT Payment_Id
FROM Payment_type
WHERE ptype = &<name="Payment Type" type="string" list="select PTYPE from Payment_type">
)
WHERE Customer_Id = &<name="Customer Id">;
```

Variables

Name	Value
Payment Type	PayPal
Customer Id	113

OK Cancel Clear

תוצאת השאילתא – רואים שה payment_id השתנה ל 4 שזה paypal בהתאם לבחירת הלקוח:



42	29/05/2025	Luxury Pavilion	7151	42	286	26	113	4
195	11/02/2025	Topaz Terrace	12931	195	204	211	113	4
145	12/10/2022	Starlight Pavil	18651	145	580	413	113	4

אילוצים

אילוץ unique על event:

זמר יכול להופיע רק במקום אחד ביום:

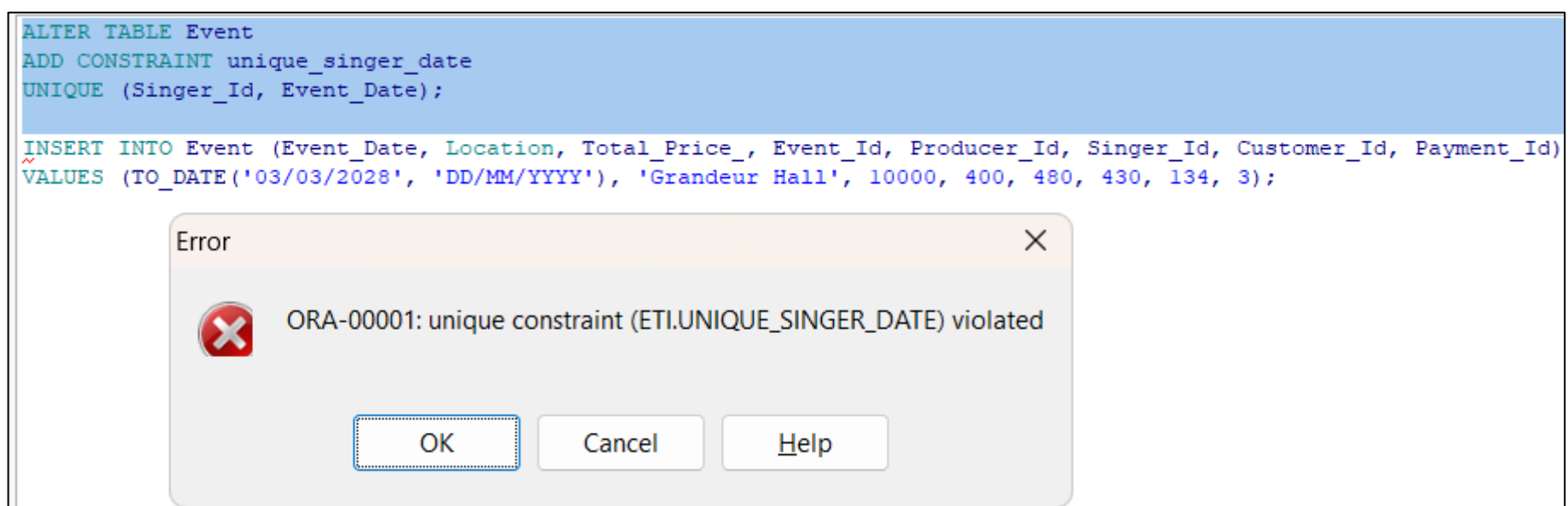
```
ALTER TABLE Event
ADD CONSTRAINT unique_singer_date
UNIQUE (Singer_Id, Event_Date);
```

```
INSERT INTO Event (Event_Date, Location, Total_Price_, Event_Id, Producer_Id,
Singer_Id, Customer_Id, Payment_Id)
VALUES (TO_DATE('03/03/2028', 'DD/MM/YYYY'), 'Grandeur Hall', 10000, 400, 480,
430, 134, 3);
```

כאן רואים שזמר 430 מופיע בתאריך 03/03/2028 ולכן כשננסה להוסיף עוד אירוע שהוא מופיע באותו תאריך נתקל בשגיאה:

		EVENT_DATE	LOCATION	TOTAL_PRICE_	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
▶	4	03/03/2028	Grandeur Hall	15002	4	238	430	257	2
	5	29/01/2021	Pearl Pavilion	11647	5	278	62	541	3
	6	23/11/2020	Pearl Pavilion	14261	6	431	411	138	1
	7	24/11/2027	Gilded Palace	14828	7	406	143	184	2
	8	27/08/2022	Topaz Terrace	14103	8	480	166	134	5
	9	26/05/2025	Moonbeam Hall	16433	9	529	392	376	3

8 7:1 eti@XE 383 rows selected in 0.266 seconds



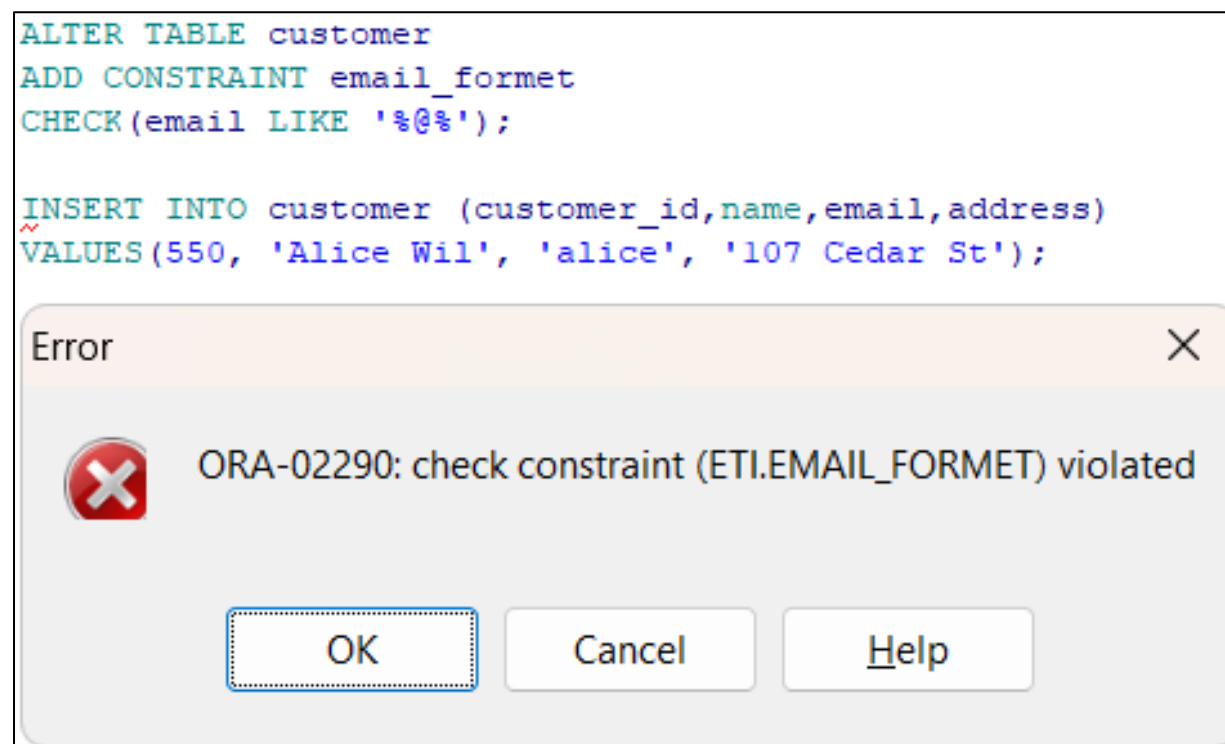
אילוץ check על customer:

מייל של לקוח חייב להכיל @:

```
ALTER TABLE customer
ADD CONSTRAINT email_format
CHECK(email LIKE '%@%');
```

```
INSERT INTO customer (customer_id,name,email,address)
VALUES(550, 'Alice Wil', 'alice', '107 Cedar St');
```

ניסינו להכניס מייל של לקוח שאינו מכיל @ ונתקלנו בשגיאה:



אילוץ default על producer:

בברירת מחדל בהוספת מפיק חדש מחירו יהיה 5000:

```
ALTER TABLE producer
MODIFY price DEFAULT 5000;
```

```
INSERT INTO producer(producer_id,producer_name)
VALUES(601,'Grace Will');
```

הכנסנו מפיק חדש ללא מחיר, וראינו שבברירת מחדל הושם לו מחיר 5000:

```
ALTER TABLE producer
MODIFY price DEFAULT 5000;

INSERT INTO producer(producer_id,producer_name)
VALUES(601,'Grace Will');

select * from producer;
```

	PRODUCER_ID	PRODUCER_NAME	PRICE
398	598	Alice Johnson	5948
399	599	Grace Williams	4924
400	600	Carol Jones	1713
401	601	Grace Will	5000

תרגיל 3:

תוכניות, פונקציות ופרוצדורות

תוכנית 1:

המקרה: זמר מסוים עבר בדיקה רפואית, והוחלט שכלי הנגינה "משולש" מזיק לו לאוזניים, ולכן ביקש שיסירו את כל השימושים של כלי זה מהאירועים בהם הוא עתיד להופיע.

התוכנית כוללת פרוצדורה ראשית שמבצעת קריאה לפונקציה ולאחריה קריאה לפרוצדורה.

ראשית נציג את הפונקציה `get_future_singer_events`:

הפונקציה מחזירה את האירועים העתידיים בהם עתיד להשתתף הזמר המבוקש.

```
CREATE OR REPLACE FUNCTION
get_future_singer_events(p_singer_id IN
Singer.Singer_Id%TYPE)
RETURN SYS_REFCURSOR
AS
    v_events SYS_REFCURSOR;
BEGIN
    OPEN v_events FOR
        SELECT Event_Id
        FROM Event
        WHERE Singer_Id = p_singer_id
        AND Event_Date >= TRUNC(SYSDATE)
        ORDER BY Event_Date;

    RETURN v_events;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No future events found for this
singer. ');
        RETURN NULL;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Unexpected error: ' ||
SQLERRM);
        RAISE;
END get_future_singer_events;
```

נציג את הפרוצדורה `: remove_instrument_from_events` הפרוצדורה מסירה את הכלי המבוקש מכל האירועים בהם עתיד הזמר להופיע (אם קיים).

```
CREATE OR REPLACE PROCEDURE
remove_instrument_from_events (p_event_cursor IN
SYS_REFCURSOR, p_instrument_id IN
Instrument.Instrument_Id%TYPE)
AS
    v_event_id Event.Event_Id%TYPE;
    v_count NUMBER := 0;
    v_instrument_name Instrument.Instrument_Name%TYPE;

    -- Custom exception
    instrument_not_found EXCEPTION;
    PRAGMA EXCEPTION_INIT(instrument_not_found, -20001);

BEGIN
    -- Check if the instrument exists
    BEGIN
        SELECT Instrument_Name INTO v_instrument_name
        FROM Instrument
        WHERE Instrument_Id = p_instrument_id;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE instrument_not_found;
    END;
```

```

LOOP
    FETCH p_event_cursor INTO v_event_id;
    EXIT WHEN p_event_cursor%NOTFOUND;

    BEGIN
        DELETE FROM Instrument_Event
        WHERE Event_Id = v_event_id
        AND Instrument_Id = p_instrument_id;

        v_count := v_count + SQL%ROWCOUNT;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            CONTINUE;
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error deleting
instrument from event: ' || SQLERRM);
            CONTINUE;

    END;
END LOOP;

    DBMS_OUTPUT.PUT_LINE(v_count || ' instances of instrument
' || v_instrument_name || ' were removed from events.');
```

```

    COMMIT;
EXCEPTION
    WHEN instrument_not_found THEN
        DBMS_OUTPUT.PUT_LINE('Instrument not found');
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error removing instrument: ' ||
SQLERRM);
        RAISE;
END remove_instrument_from_events;
```


נציג את הפרוצדורה :manage_singer_instrument
הפרוצדורה משמשת כתוכנית ראשית להרצת הפונקציה והפרוצדורה.

```
CREATE OR REPLACE PROCEDURE
manage_singer_instrument(p_singer_id IN
Singer.Singer_Id%TYPE, p_instrument_id IN
Instrument.Instrument_Id%TYPE)
AS
    v_events SYS_REFCURSOR;

    -- Record type for storing singer details
    TYPE r_singer_info IS RECORD (
        name Singer.Sname%TYPE,
        price Singer.Price%TYPE
    );
    v_singer_info r_singer_info;
BEGIN
    -- Get singer details
    BEGIN
        SELECT Sname, Price
        INTO v_singer_info.name, v_singer_info.price
        FROM Singer
        WHERE Singer_Id = p_singer_id;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20002, 'Singer not
found');
    END;

    DBMS_OUTPUT.PUT_LINE('Managing events for singer ' ||
v_singer_info.name ||
                        ' (Price: ' || v_singer_info.price
|| ')');

    -- Get list of future events for the singer
    v_events := get_future_singer_events(p_singer_id);

    IF v_events IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('No future events found for this
singer. ');
        RETURN;
    END IF;
```



```

1 begin
2     -- Call the procedure
3     manage_singer_instrument(p_singer_id => :p_singer_id,
4                             p_instrument_id => :p_instrument_id);
5 end;

```

	Variable	Type	Value
<input checked="" type="checkbox"/>	p_singer_id	Integer	430
<input checked="" type="checkbox"/>	p_instrument_id	Integer	16
*			

1:1 eti@XE Executed in 0.016 seconds

תוצאת ההרצה:

Managing events for singer Jane Wilson (Price: 7223)
 1 instances of instrument Triangle were removed from events.
 Operation completed successfully.

4:1 eti@XE Executed in 0.016 seconds

select * from singer where singer_id=430;

	SNAME	SINGER_ID	PRICE
1	Jane Wilson	430	7223

כלי מספר 16 הוסר מהאירוע:

select * from instrument_event;

	INSTRUMENT_ID	EVENT_ID
116	15	1
12	2	1
1	1	1
2	1	2
294	38	4
231	28	4
330	43	5
127	16	6

5:1 eti@XE 0:02

תוכנית 2:

המקרה: בעקבות המצב הכלכלי הקשה, החליטו בחברת המוזיקה לצ'פר את המפיקים בהתאם לכמות האירועים בהם לקחו חלק. בעקבות צ'פור שכר המפיקים יש צורך לעדכן את הסכום הכולל לאירוע בטבלת האירועים.

התוכנית כוללת טסט ראשי שמבצע קריאה לפרוצדורה ולאחריה קריאה לפונקציה.

ראשית נציג את הפרוצדורה `UpdateProducerPrices`:

הפרוצדורה מעדכנת את מחירי המפיקים בהתאם לכמות האירועים בהם לקחו חלק.

```
CREATE OR REPLACE PROCEDURE UpdateProducerPrices AS
  -- Declare record type for producer details
  TYPE producer_rec_type IS RECORD (
    Producer_Id Producer.Producer_Id%TYPE,
    Price Producer.Price%TYPE
  );

  -- Declare cursor for producers
  CURSOR producer_cur IS
    SELECT Producer_Id, Price FROM Producer;

  -- Declare variable to hold producer record
  producer_rec producer_rec_type;

  -- Declare variables for event count and price increase
  v_event_count NUMBER;
  v_price_increase NUMBER;
  v_producers_count NUMBER := 0;
```

```

BEGIN
    -- Open explicit cursor
    OPEN producer_cur;

    -- Loop through each producer
    LOOP
        FETCH producer_cur INTO producer_rec;
        EXIT WHEN producer_cur%NOTFOUND;

        -- Count the number of events for each producer
        SELECT COUNT(*) INTO v_event_count
        FROM Event
        WHERE Producer_Id = producer_rec.Producer_Id;

        -- Calculate the price increase percentage (capped at 10%)
        v_price_increase := LEAST(v_event_count, 10);

        -- Update the producer's price
        UPDATE Producer
        SET Price = Price * (1 + (v_price_increase / 100))
        WHERE Producer_Id = producer_rec.Producer_Id;

        -- Print update information
        DBMS_OUTPUT.PUT_LINE('Producer ID: ' ||
producer_rec.Producer_Id ||
                                ', Events: ' || v_event_count ||
                                ', Price increase: ' ||
v_price_increase || '%');
        v_producers_count:=v_producers_count+1;
    END LOOP;

    -- Close explicit cursor
    CLOSE producer_cur;

    -- Print confirmation message
    DBMS_OUTPUT.PUT_LINE(v_producers_count || ' producers prices
updated successfully.');
```

```

COMMIT;

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error updating producer prices: '
|| SQLERRM);
END UpdateProducerPrices;
```

נציג את הפונקציה `:UpdateTotalEventCosts`

הפונקציה מעדכנת את הסכום הכולל לאירוע, ומחזירה את סכום כל האירועים.

```
CREATE OR REPLACE FUNCTION UpdateTotalEventCosts
RETURN NUMBER
IS
    CURSOR event_cursor IS
        SELECT Event_Id FROM Event;

    event_id_ NUMBER;
    instrument_cost NUMBER;
    total_cost NUMBER;
    overall_total_cost NUMBER := 0;
BEGIN
    OPEN event_cursor;

    LOOP
        FETCH event_cursor INTO event_id_;
        EXIT WHEN event_cursor%NOTFOUND;

        -- Calculate the cost of instruments for the event
        BEGIN
            SELECT COALESCE(SUM(i.Price), 0)
            INTO instrument_cost
            FROM Instrument_Event ie
            JOIN Instrument i ON ie.Instrument_Id = i.Instrument_Id
            WHERE ie.Event_Id = event_id_;
        EXCEPTION
            WHEN OTHERS THEN
                instrument_cost := 0;
        END;

        -- Calculate the total cost of the event
        BEGIN
            SELECT s.Price + p.Price + instrument_cost
            INTO total_cost
            FROM Event e
            JOIN Singer s ON e.Singer_Id = s.Singer_Id
            JOIN Producer p ON e.Producer_Id = p.Producer_Id
            WHERE e.Event_Id = event_id_;
        EXCEPTION
            WHEN OTHERS THEN
                total_cost := 0;
        END;

        overall_total_cost := overall_total_cost + total_cost;
    END LOOP;

    RETURN overall_total_cost;
END;
```

```

-- Update the total cost in the Event table
UPDATE Event
SET Total_price_ = total_cost
WHERE Event_Id = event_id_;

-- Add to overall total cost
overall_total_cost := overall_total_cost + total_cost;

END LOOP;

CLOSE event_cursor;

RETURN overall_total_cost;
END UpdateTotalEventCosts;

```

נציג את הטסט הראשי:

הטסט משמש כתוכנית ראשית להרצת הפונקציה והפרוצדורה.

```

DECLARE
    total_cost NUMBER;
BEGIN
    UpdateProducerPrices;
    total_cost := UpdateTotalEventCosts;
    DBMS_OUTPUT.PUT_LINE('Total cost of all events: ' ||
total_cost);
END;

```

נראה נכונות:

בדוגמאות נתייחס למפיק מסויים.

כמות האירועים בהם השתתף כל מפיק:

מחיר מפיק לפני ההרצה:

```
select producer_id, price from producer where producer_id=303;
```

	PRODUCER_ID	PRICE
1	303	4863

1:21 eti@XE 1 row selected in 0.016 seconds

מחירי מפיק אחרי ההרצה:

```
select producer_id, price from producer where producer_id=303;
```

	PRODUCER_ID	PRICE
1	303	5058

2:21 eti@XE 1 row selected in 0.032 seconds

SQL Window - select producer_id, co

SQL Output Statistics

```
select producer_id, count(*)  
from event  
group by producer_id
```

	PRODUCER_ID	COUNT(*)
1	504	1
2	555	3
3	407	1
4	544	1
5	310	1
6	537	2
7	244	1
8	423	2
9	335	1
10	280	1
11	574	2
12	477	1
13	380	2
14	498	3
15	303	4

רואים פה את המחיר הכולל של האירוע, ואת המספר של המפיק לפני ההרצה:

```
select * from event where producer_id=303;  
select price from producer where producer_id=303;
```

	EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
1	04/06/2026	Blissful Palace	13471	205	303	435	170	

נראה שלאחר ההרצה הסכום הכולל השתנה:

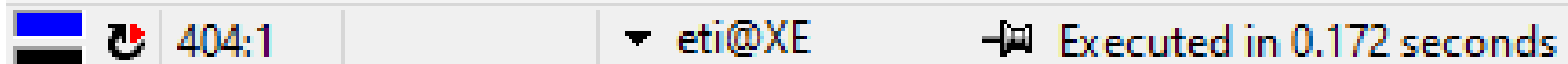
```
select * from event where producer_id=303;  
select producer_id, price from producer where producer_id=303;
```

	EVENT_DATE	LOCATION	TOTAL_PRICE	EVENT_ID	PRODUCER_ID	SINGER_ID	CUSTOMER_ID	PAYMENT_ID
1	04/06/2026	Blissful Palace	13666	205	303	435	170	

המחיר של המפיק הוכפל ב1.04 מכיוון שהשתתף ב4 אירועים.
המחיר של המפיק עלה ב195 ש.
ואכן גם המחיר הכולל של האירוע עלה ב195 ש.

מוצגת פה ההדפסה כמובן ארוכה יותר אבל לא נלאה.....

```
Producer ID: 597, Events: 0, Price increase: 0%
Producer ID: 598, Events: 2, Price increase: 2%
Producer ID: 599, Events: 2, Price increase: 2%
Producer ID: 600, Events: 0, Price increase: 0%
Producer ID: 601, Events: 0, Price increase: 0%
401 producers prices updated successfully.
Total cost of all events: 5010407
```

 404:1 eti@XE Executed in 0.172 seconds