

צילום וידאו

אירועים

מגישות:

שילת שרון 325541100

לאה דרך 214202921

תוכן עניינים:

3.....	<u>תיאור המערכת</u>
4.....	<u>ERD תרשים</u>
5.....	<u>DSD תרשים</u>
6.....	<u>קבצי SQL</u>
6.....	Create קובץ
7.....	Drop קובץ
7.....	Insert קובץ
10.....	SelectAll קובץ
11.....	<u>Data Generator</u>
11.....	טבלת Client
11.....	טבלת Event
12.....	טבלת CustomerOrder
12.....	טבלת Payment
13.....	<u>Text Importer</u>
13.....	טבלת Professional
14.....	<u>Programming</u>
14.....	טבלת Include
15.....	<u>גיבוי ושחזור</u>
15.....	שיטת SQL Insert
16.....	<u>desc פקודת</u>
18.....	שאליות
18.....	Select שאליות
22.....	Update שאליות
24.....	Delete שאליות
27.....	<u>אילוצים</u>
30.....	<u>שאליות עם פרמטרים</u>
33.....	<u>פונקציות</u>
37.....	<u>פרוצדורות</u>
41.....	<u>תוכניות ראשיות</u>
45.....	<u>אינטגרציה</u>
45.....	Reverse engineering
48.....	Integrated ERD
50.....	אינטגרציה בבסיס הנתונים
53.....	<u>views</u>

תיאור המערכת

ישויות:

Profession - מקצוע (צלם או מסריט ווידאו)

שדות: מספר מזהה, שם המקצוע.

Professional - בעל מקצוע. יש לו מקצוע יחיד.

שדות: מספר מזהה, שם פרטי, שם משפחה, טלפון, מייל, שנת התחלה, מחיר.

Client - לקוח

שדות: מספר מזהה, שם פרטי, שם משפחה, טלפון, מייל

PaymentMethod - שיטת תשלום. (אשראי/ מזומן/ העברה בנקאית/ צ'ק)

שדות: מספר מזהה, סוג תשלום.

Payment - תשלום. לתשלום יש אמצעי תשלום יחיד.

שדות: מספר מזהה, תאריך תשלום, סכום לתשלום.

Event - אירוע

שדות: מספר מזהה, תאריך אירוע, מקום.

CustomerOrder – הזמנה. מורכבת מכמה אנשי מקצוע וכמה תשלומים.

שייכת לאירוע יחיד ולקוח יחיד.

שדות: מספר מזהה, תאריך הזמנה.

סוגי קשרים:

Work - קשר בין מקצוע לבעל מקצוע. לכל בעל מקצוע יכול להיות מקצוע יחיד.

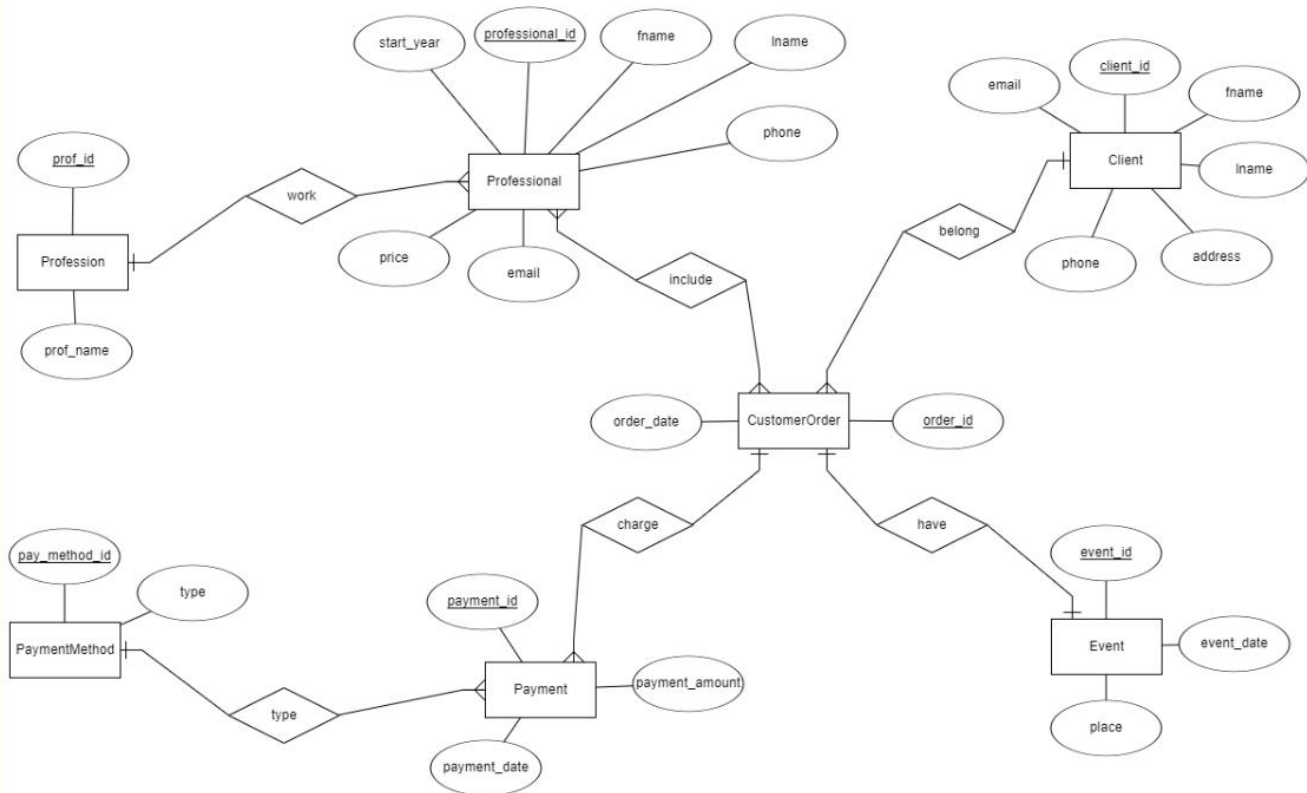
Belong - קשר בין הזמנה ללקוח. להזמנה יש לקוח אחד. ללקוח יכולות להיות כמה הזמנות.

Have - קשר בין הזמנה לאירוע. לכל הזמנה אירוע יחיד ולכל אירוע הזמנה יחידה.

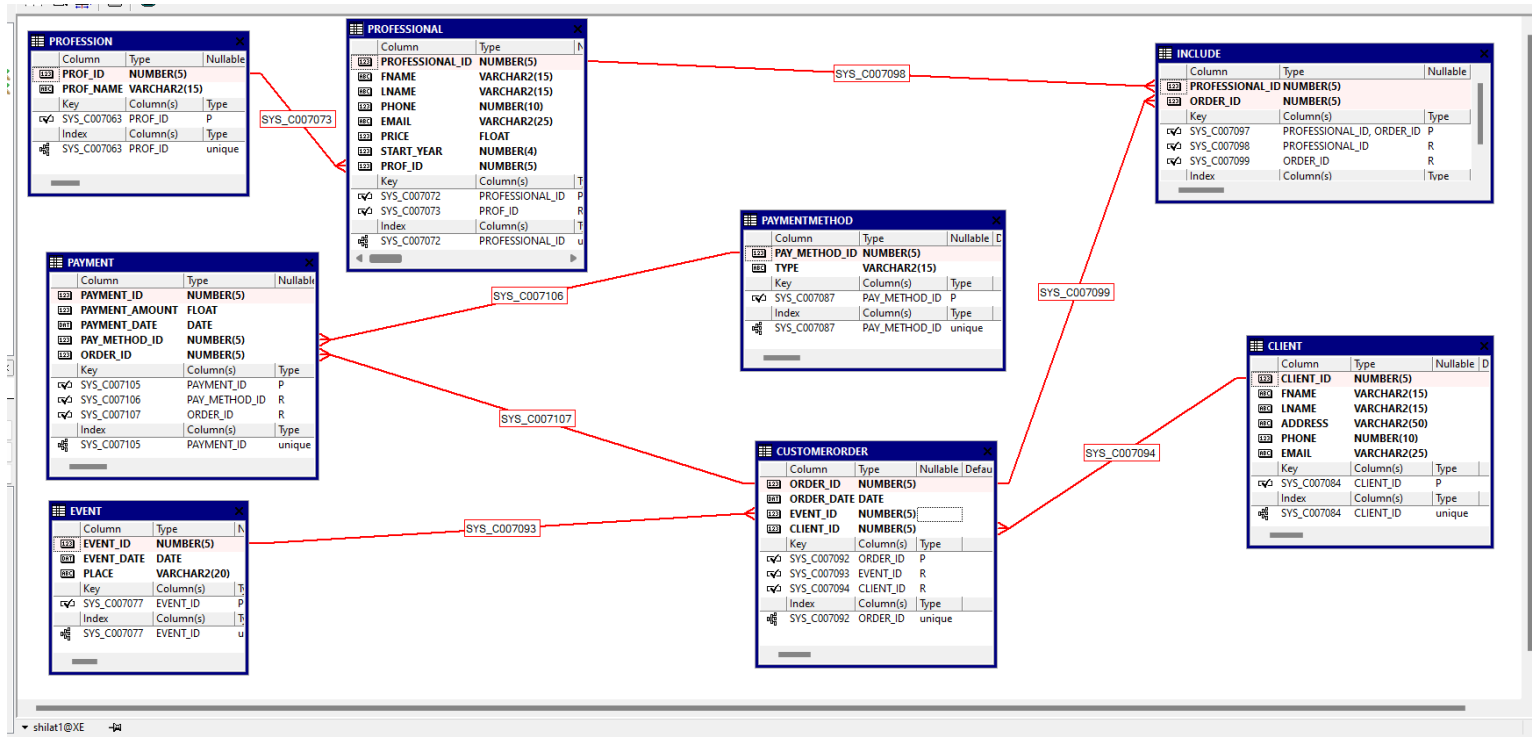
Charge - קשר בין הזמנה לתשלום. לכל הזמנה יתכנו מספר תשלומים. כל תשלום שייך להזמנה יחידה.

Type - קשר בין תשלום לסוג תשלום. לכל תשלום שיטת תשלום יחידה. שיטת תשלום יכולה להיות משותפת למספר תשלומים.

תרשים ERD



תרשים DSD



קובץ Create

```
CREATE TABLE Profession
(
    prof_id INT NOT NULL,
    prof_name INT NOT NULL,
    PRIMARY KEY (prof_id)
);

CREATE TABLE Professional
(
    professional_id INT NOT NULL,
    fname INT NOT NULL,
    lname INT NOT NULL,
    phone INT NOT NULL,
    email INT NOT NULL,
    price INT NOT NULL,
    start_year INT NOT NULL,
    prof_id INT NOT NULL,
    PRIMARY KEY (professional_id),
    FOREIGN KEY (prof_id) REFERENCES Profession(prof_id)
);

CREATE TABLE Event
(
    event_id INT NOT NULL,
    event_date INT NOT NULL,
    place INT NOT NULL,
    PRIMARY KEY (event_id)
);

CREATE TABLE Client
(
    client_id INT NOT NULL,
    fname INT NOT NULL,
    lname INT NOT NULL,
    address INT NOT NULL,
    phone INT NOT NULL,
    email INT NOT NULL,
    PRIMARY KEY (client_id)
);

CREATE TABLE PaymentMethod
(
    pay_method_id INT NOT NULL,
    type INT NOT NULL,
    PRIMARY KEY (pay_method_id)
);

CREATE TABLE CustomerOrder
(
    order_id INT NOT NULL,
    order_date INT NOT NULL,
    event_id INT NOT NULL,
    client_id INT NOT NULL,
    PRIMARY KEY (order_id),
```

```

    FOREIGN KEY (event_id) REFERENCES Event(event_id),
    FOREIGN KEY (client_id) REFERENCES Client(client_id)
);

CREATE TABLE include
(
    professional_id INT NOT NULL,
    order_id INT NOT NULL,
    PRIMARY KEY (professional_id, order_id),
    FOREIGN KEY (professional_id) REFERENCES
Professional(professional_id),
    FOREIGN KEY (order_id) REFERENCES CustomerOrder(order_id)
);

CREATE TABLE Payment
(
    payment_id INT NOT NULL,
    payment_amount INT NOT NULL,
    payment_date INT NOT NULL,
    pay_method_id INT NOT NULL,
    order_id INT NOT NULL,
    PRIMARY KEY (payment_id),
    FOREIGN KEY (pay_method_id) REFERENCES
PaymentMethod(pay_method_id),
    FOREIGN KEY (order_id) REFERENCES CustomerOrder(order_id)
);

```

קובץ Drop

```

DROP TABLE Include;
DROP TABLE Payment;
DROP TABLE CustomerOrder;
DROP TABLE Event;
DROP TABLE Client;
DROP TABLE PaymentMethod;
DROP TABLE Professional;
DROP TABLE Profession;

```

קובץ Insert

```

insert into PROFESSION (prof_id, prof_name)
values (0, 'photographer');
insert into PROFESSION (prof_id, prof_name)
values (1, 'video');

insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)
values (0, ' John', ' Smith', 511652243, ' john.smith@gmail.com',
37700, 2020, 0);
insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)
values (1, ' Paul', ' Smith', 583118533, ' paul.smith@gmail.com',
14600, 2015, 0);
insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)

```

```

values (2, 'Paul', 'Martinez', 511999596, '
paul.martinez@gmail.com', 7500, 2022, 0);
insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)
values (3, 'Katie', 'Miller', 573072267, 'katie.miller@gmail.com',
41400, 2005, 1);
insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)
values (4, 'Sara', 'Hernandez', 506837826, '
sara.hernandez@gmail.com', 25700, 2009, 0);
insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)
values (5, 'Paul', 'Williams', 531004594, '
paul.williams@gmail.com', 19100, 2015, 1);
insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)
values (6, 'Anna', 'Garcia', 578516792, 'anna.garcia@gmail.com',
41200, 2022, 0);
insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)
values (7, 'Alex', 'Martinez', 514172647, '
alex.martinez@gmail.com', 16200, 2000, 1);
insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)
values (8, 'Katie', 'Smith', 571446153, 'katie.smith@gmail.com',
12200, 2018, 0);
insert into PROFESSIONAL (professional_id, fname, lname, phone,
email, price, start_year, prof_id)
values (9, 'Alex', 'Williams', 525793180, '
alex.williams@gmail.com', 33100, 2017, 0);

insert into PAYMENTMETHOD (pay_method_id, type)
values (0, 'Credit Card');
insert into PAYMENTMETHOD (pay_method_id, type)
values (1, 'Cash');
insert into PAYMENTMETHOD (pay_method_id, type)
values (2, 'PayPal');
insert into PAYMENTMETHOD (pay_method_id, type)
values (3, 'Bank Transfer');

insert into CLIENT (client_id, fname, lname, address, phone, email)
values (0, 'Arnold', 'Hornsby', '69 DeVita Ave', 582229278,
'arnold.hornsby@ads.com');
insert into CLIENT (client_id, fname, lname, address, phone, email)
values (1, 'Bruce', 'Baranski', '76 Robert Street', 533999149,
'bruce@hencie.com');
insert into CLIENT (client_id, fname, lname, address, phone, email)
values (2, 'Shannon', 'Weisz', '78 Kirsten Street', 586353849,
'shannon@typhoon.com');
insert into CLIENT (client_id, fname, lname, address, phone, email)
values (3, 'Bryan', 'Snipes', '81 Whitford Road', 593892633,
'bryans@spinnakerexplorati');
insert into CLIENT (client_id, fname, lname, address, phone, email)
values (4, 'Rip', 'Palmer', '836 Pablo Street', 577262923,
'r.palmer@jsa.dk');
insert into CLIENT (client_id, fname, lname, address, phone, email)
values (5, 'Rickie', 'Lapointe', '32 Child Blvd', 549347199,
'rickie.lapointe@vitacostc');
insert into CLIENT (client_id, fname, lname, address, phone, email)
values (6, 'Luis', 'Saucedo', '30 Dionne Road', 549629835,
'luis.saucedo@scheringplou');

```



```

insert into CLIENT (client_id, fname, lname, address, phone, email)
values (7, 'Robbie', 'Santa Rosa', '65 Maury Road', 547235857,
'robbie.santarosa@kwraf.co');
insert into CLIENT (client_id, fname, lname, address, phone, email)
values (8, 'Merle', 'Ramirez', '19 Byrd Ave', 545275812,
'merler@consultants.uk');
insert into CLIENT (client_id, fname, lname, address, phone, email)
values (9, 'Patricia', 'Tolkan', '72nd Street', 532817583,
'patricia@egroup.br');

insert into EVENT (event_id, event_date, place)
values (0, to_date('07-12-2018', 'dd-mm-yyyy'), 'Banquet Hall');
insert into EVENT (event_id, event_date, place)
values (1, to_date('21-10-2011', 'dd-mm-yyyy'), 'Exhibition Hall');
insert into EVENT (event_id, event_date, place)
values (2, to_date('21-01-2020', 'dd-mm-yyyy'), 'Event Arena');
insert into EVENT (event_id, event_date, place)
values (3, to_date('27-01-2009', 'dd-mm-yyyy'), 'Exhibition Hall');
insert into EVENT (event_id, event_date, place)
values (4, to_date('14-04-2000', 'dd-mm-yyyy'), 'Grand Hall');
insert into EVENT (event_id, event_date, place)
values (5, to_date('26-10-2005', 'dd-mm-yyyy'), 'Grand Hall');
insert into EVENT (event_id, event_date, place)
values (6, to_date('17-02-2012', 'dd-mm-yyyy'), 'Banquet Hall');
insert into EVENT (event_id, event_date, place)
values (7, to_date('01-05-2003', 'dd-mm-yyyy'), 'Event Arena');
insert into EVENT (event_id, event_date, place)
values (8, to_date('27-02-2014', 'dd-mm-yyyy'), 'Grand Hall');
insert into EVENT (event_id, event_date, place)
values (9, to_date('05-08-2004', 'dd-mm-yyyy'), 'Event Arena');

insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (289, to_date('11-10-2022', 'dd-mm-yyyy'), 177, 59);
insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (290, to_date('06-01-2020', 'dd-mm-yyyy'), 224, 26);
insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (291, to_date('22-04-2024', 'dd-mm-yyyy'), 343, 220);
insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (292, to_date('03-01-2020', 'dd-mm-yyyy'), 314, 205);
insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (293, to_date('25-03-2020', 'dd-mm-yyyy'), 383, 68);
insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (294, to_date('20-08-2020', 'dd-mm-yyyy'), 121, 204);
insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (295, to_date('27-09-2022', 'dd-mm-yyyy'), 21, 351);
insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (296, to_date('23-03-2023', 'dd-mm-yyyy'), 160, 230);
insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (297, to_date('18-04-2022', 'dd-mm-yyyy'), 155, 27);
insert into CUSTOMERORDER (order_id, order_date, event_id, client_id)
values (298, to_date('11-01-2021', 'dd-mm-yyyy'), 22, 109);

insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (0, 45000, to_date('30-06-2021', 'dd-mm-yyyy'), 2, 0);
insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (1, 24000, to_date('05-04-2020', 'dd-mm-yyyy'), 1, 1);
insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (2, 25000, to_date('27-12-2022', 'dd-mm-yyyy'), 2, 2);

```

```

insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (3, 43000, to_date('03-11-2020', 'dd-mm-yyyy'), 2, 3);
insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (4, 24000, to_date('09-05-2021', 'dd-mm-yyyy'), 1, 4);
insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (5, 16000, to_date('29-08-2023', 'dd-mm-yyyy'), 1, 5);
insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (6, 17000, to_date('17-06-2020', 'dd-mm-yyyy'), 1, 6);
insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (7, 3000, to_date('21-09-2020', 'dd-mm-yyyy'), 0, 7);
insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (8, 1000, to_date('22-04-2022', 'dd-mm-yyyy'), 3, 8);
insert into PAYMENT (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
values (9, 20000, to_date('27-07-2021', 'dd-mm-yyyy'), 3, 9);

insert into INCLUDE (professional_id, order_id)
values (0, 13);
insert into INCLUDE (professional_id, order_id)
values (0, 87);
insert into INCLUDE (professional_id, order_id)
values (1, 162);
insert into INCLUDE (professional_id, order_id)
values (2, 285);
insert into INCLUDE (professional_id, order_id)
values (5, 337);
insert into INCLUDE (professional_id, order_id)
values (7, 236);
insert into INCLUDE (professional_id, order_id)
values (10, 384);
insert into INCLUDE (professional_id, order_id)
values (11, 380);
insert into INCLUDE (professional_id, order_id)
values (11, 392);
insert into INCLUDE (professional_id, order_id)
values (14, 108);

```

קובץ SelectAll

```

Select * from Profession;
Select count(*) from Profession;
Select * from Professional;
Select count(*) from Professional;
Select * from PaymentMethod;
Select count(*) from PaymentMethod;
Select * from Client;
Select count(*) from Client;
Select * from Event;
Select count(*) from Event;
Select * from CustomerOrder;
Select count(*) from CustomerOrder;
Select * from Payment;

```

```

Select count(*) from Payment;
Select * from Include;
Select count(*) from Include;

```

Data Generator

טבלת Client

CLIENT				
Owner	Table	Number of records		
SHILAT1	CLIENT	10..20		
Name	Type	Size		Data
CLIENT_ID	NUMBER	5		Sequence(0, [Inc], [WithinParent])
FNAME	VARCHAR2	15		FirstName
LNAME	VARCHAR2	15		LastName
ADDRESS	VARCHAR2	50		Address1
PHONE	NUMBER	10		'05'[11111111]
EMAIL	VARCHAR2	25		Email
*				

טבלת Event

EVENT				
Owner	Table	Number of records		
SHILAT1	EVENT	400		
Name	Type	Size		Data
EVENT_ID	NUMBER	5		Sequence(0, [Inc], [WithinParent])
EVENT_DATE	DATE			Random(1/1/2000, 20/5/2024)
PLACE	VARCHAR2	20		List('Conference Center', 'Grand Hall', 'Exhibition Hall', 'Banquet Hall', 'Event Arena')
*				

טבלת CustomerOrder

CUSTOMERORDER				
Owner	Table		Number of records	
SHILAT1	CUSTOMERORDER		400	
Name	Type	Size		Data
ORDER_ID	NUMBER	5		Sequence(0, [Inc], [WithinParent])
ORDER_DATE	DATE			Random(1/1/2000,20/5/2024)
EVENT_ID	NUMBER	5		List(select event_id from Event)
CLIENT_ID	NUMBER	5		List(select client_id from Client)
*				

טבלת Payment

PAYMENT				
Owner	Table		Number of records	
SHILAT1	PAYMENT		400	
Name	Type	Size		Data
PAYMENT_ID	NUMBER	5		Sequence(0, [Inc], [WithinParent])
PAYMENT_AMOUNT	FLOAT	22		List(select price from Professional)
PAYMENT_DATE	DATE			Random(1/1/2020, 20/5/2024)
PAY_METHOD_ID	NUMBER	5		List(select pay_method_id from PaymentMethod)
ORDER_ID	NUMBER	5		Sequence(0, [Inc], [WithinParent])
*				

Text Importer

שבילת Professional

Data from Textfile | Data to Oracle

File Data

```
0, Chris, Davis, 0511827863, chris@gmail.com, 8000, 2000, 1
1, Jane, Martinez, 0529914114, jane@gmail.com, 21000, 2006, 0
2, Paul, Jones, 0569433112, paul@gmail.com, 34000, 2005, 0
3, Sara, Williams, 0539177984, sara@gmail.com, 12000, 2000, 1
4, Anna, Davis, 0522654295, anna@gmail.com, 44000, 2009, 0
5, Alex, Brown, 0597275161, alex@gmail.com, 31000, 2022, 0
6, Anna, Hernandez, 0547734795, anna@gmail.com, 23000, 2019, 1
7, Alex, Brown, 0568683737, alex@gmail.com, 50000, 2011, 0
8, Anna, Hernandez, 0519326032, anna@gmail.com, 45000, 2021, 0
9, Anna, Brown, 0580032827, anna@gmail.com, 6000, 2017, 0
10, Mike, Davis, 0559760186, mike@gmail.com, 5000, 2022, 1
11, Alex, Smith, 0591151118, alex@gmail.com, 45000, 2012, 0
12, Chris, Williams, 0520403581, chris@gmail.com, 6000, 2013, 1
13, Mike, Brown, 0588154039, mike@gmail.com, 42000, 2008, 1
14, Paul, Jones, 0541948481, paul@gmail.com, 28000, 2019, 0
15, Anna, Martinez, 0541915479, anna@gmail.com, 33000, 2002, 0
16, Anna, Smith, 0510042569, anna@gmail.com, 10000, 2014, 0
17, Paul, Martinez, 0544797765, paul@gmail.com, 50000, 2003, 1
18, Anna, Martinez, 0546595330, anna@gmail.com, 46000, 2004, 1
19, Chris, Davis, 0507191431, chris@gmail.com, 9000, 2004, 1
20, Alex, Brown, 0563595948, alex@gmail.com, 7000, 2013, 0
21, Paul, Brown, 0515483487, paul@gmail.com, 31000, 2002, 1
22, John, Miller, 0552787050, john@gmail.com, 37000, 2014, 0
23, Paul, Martinez, 0500842037, paul@gmail.com, 19000, 2006, 0
24, Chris, Jones, 0511700281, chris@gmail.com, 18000, 2017, 1
25, Katie, Hernandez, 0586566307, katie@gmail.com, 30000, 2021, 1
```

Configuration

General

Fieldcount: 8

☒ End at line-end

☐ Name in header

☒ Skip empty lines

Quote character: "

Comment line: #

Import lines: 1 ..

Field1 (+0..")
Field2 (+0..")
Field3 (+0..")
Field4 (+0..")
Field5 (+0..")
Field6 (+0..")
Field7 (+0..")
Field8 (+0..")

Field Start

☐ Relative position

☐ Absolute position

☐ Character

Field End

☐ Length

☐ Character

Filter:

Result Preview

1	2	3	4	5	6	7	8
0	Chris	Davis	0511827863	chris@gmail.com	8000	2000	1
1	Jane	Martinez	0529914114	jane@gmail.com	21000	2006	0

Import Import to Script Close shilat1@XE professionalDatabase.txt loaded, 24 KB

Data from Textfile | Data to Oracle

General

Owner: Table

Table: PROFESSIONAL

Clear Table

Commit every...: 0

☐ Overwrite duplicates

☒ Ignore duplicates

Initializing Script

Finalizing Script

Fields

Field1 -> PROFESSIONAL_ID (NUMBER)

Field2 -> FNAME (VARCHAR2)

Field3 -> LNAME (VARCHAR2)

Field4 -> PHONE (NUMBER)

Field5 -> EMAIL (VARCHAR2)

Field6 -> PRICE (FLOAT)

Field7 -> START_YEAR (NUMBER)

Field8 -> PROF_ID (NUMBER)

Field: PROF_ID (NUMBER)

Fieldtype: Number

Create SQL

SQL function

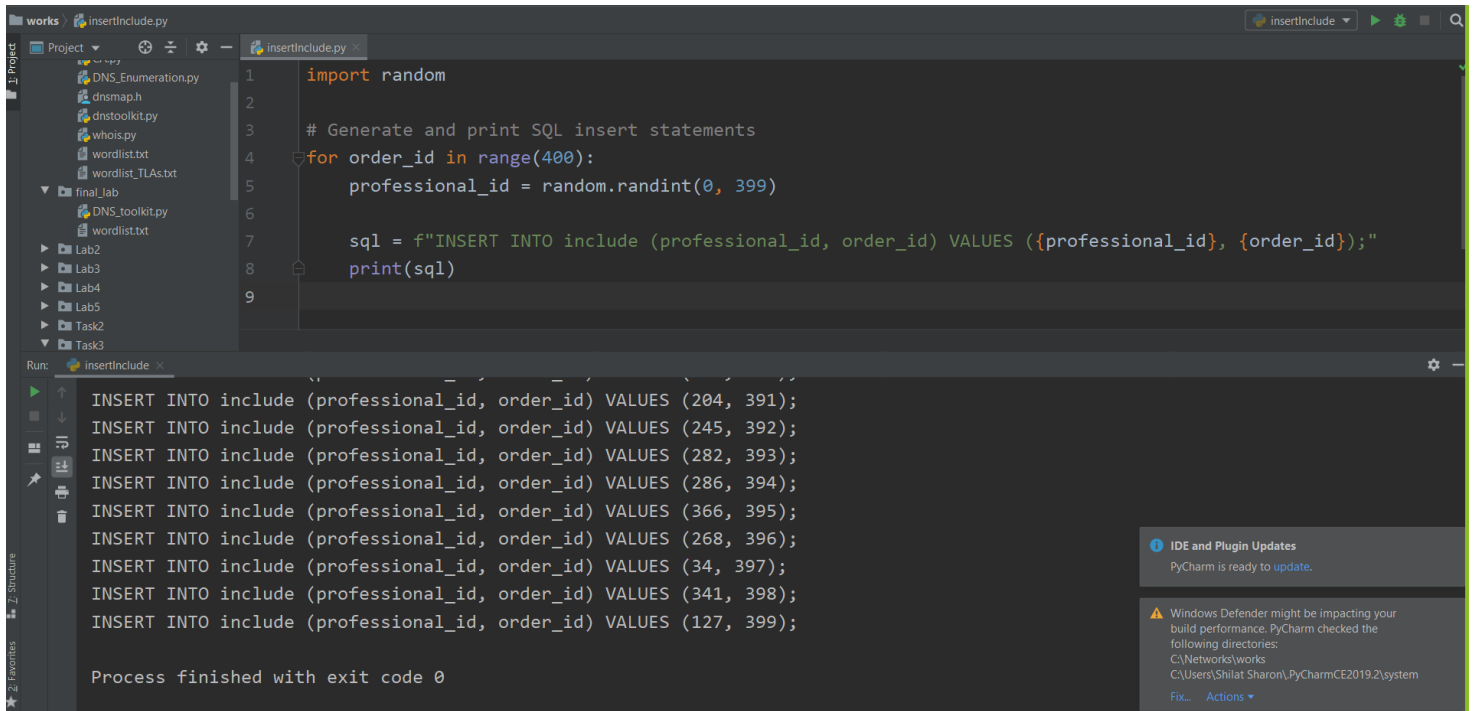
additional Oracle processing, for example: substr(4, 1, 20)

Result Preview

1	2	3	4	5	6	7	8
0	Chris	Davis	0511827863	chris@gmail.com	8000	2000	1
1	Jane	Martinez	0529914114	jane@gmail.com	21000	2006	0

Programming

טבלת Include



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `insertInclude.py`. The script imports the `random` module and generates 400 random SQL insert statements. The output of the script is shown in the Run console at the bottom, displaying a sample of the generated statements. The Run console also shows a notification for IDE and Plugin updates and a warning about Windows Defender potentially impacting build performance.

```
1 import random
2
3 # Generate and print SQL insert statements
4 for order_id in range(400):
5     professional_id = random.randint(0, 399)
6
7     sql = f"INSERT INTO include (professional_id, order_id) VALUES ({professional_id}, {order_id});"
8     print(sql)
9
```

Run: insertInclude

```
INSERT INTO include (professional_id, order_id) VALUES (204, 391);
INSERT INTO include (professional_id, order_id) VALUES (245, 392);
INSERT INTO include (professional_id, order_id) VALUES (282, 393);
INSERT INTO include (professional_id, order_id) VALUES (286, 394);
INSERT INTO include (professional_id, order_id) VALUES (366, 395);
INSERT INTO include (professional_id, order_id) VALUES (268, 396);
INSERT INTO include (professional_id, order_id) VALUES (34, 397);
INSERT INTO include (professional_id, order_id) VALUES (341, 398);
INSERT INTO include (professional_id, order_id) VALUES (127, 399);

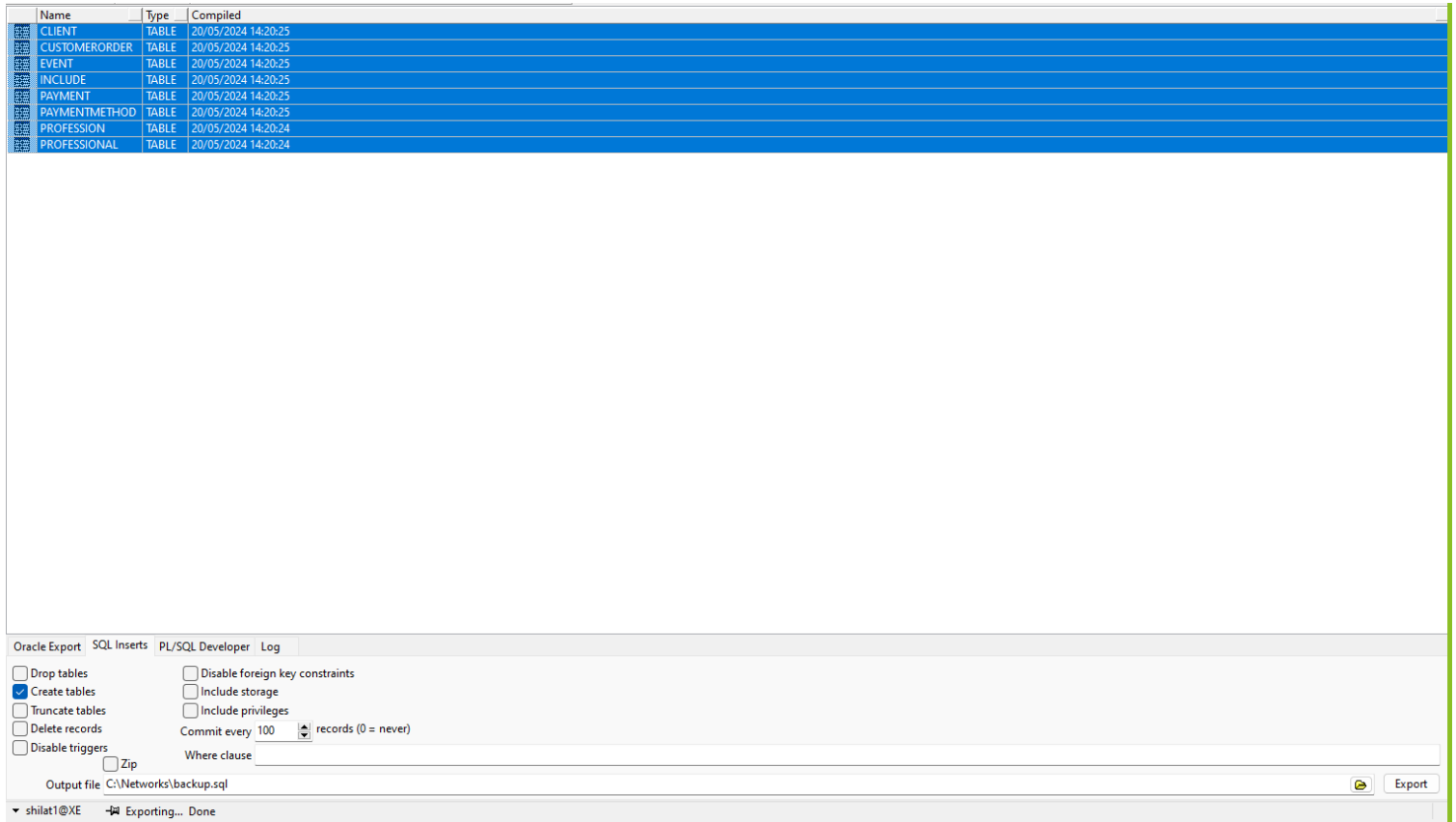
Process finished with exit code 0
```

IDE and Plugin Updates
PyCharm is ready to [update](#).

Windows Defender might be impacting your build performance. PyCharm checked the following directories:
C:\Networks\works
C:\Users\Shilat Sharon\PyCharmCE2019.2\system
[Fix...](#) [Actions](#)

גיבוי ושחזור

שיטת SQL Insert



פקודת desc

```
Connected to Oracle Database 11g Express Edition Release 11.2.0.2.0  
Connected as shilat1@XE
```

```
SQL> desc Profession
```

Name	Type	Nullable	Default	Comments
PROF_ID	NUMBER(5)			
PROF_NAME	VARCHAR2(15)			

```
SQL> desc Professional
```

Name	Type	Nullable	Default	Comments
PROFESSIONAL_ID	NUMBER(5)			
FNAME	VARCHAR2(15)			
LNAME	VARCHAR2(15)			
PHONE	NUMBER(10)			
EMAIL	VARCHAR2(25)			
PRICE	FLOAT			
START_YEAR	NUMBER(4)			
PROF_ID	NUMBER(5)			

```
SQL> desc Event
```

Name	Type	Nullable	Default	Comments
EVENT_ID	NUMBER(5)			
EVENT_DATE	DATE			
PLACE	VARCHAR2(20)			

```
SQL> desc Client
```

Name	Type	Nullable	Default	Comments
CLIENT_ID	NUMBER(5)			
FNAME	VARCHAR2(15)			
LNAME	VARCHAR2(15)			
ADDRESS	VARCHAR2(50)			
PHONE	NUMBER(10)			
EMAIL	VARCHAR2(25)			

```
SQL> desc PaymentMethod
```

Name	Type	Nullable	Default	Comments
PAY_METHOD_ID	NUMBER(5)			
TYPE	VARCHAR2(15)			

SQL> desc CustomerOrder

Name	Type	Nullable	Default	Comments
ORDER_ID	NUMBER(5)			
ORDER_DATE	DATE			
EVENT_ID	NUMBER(5)			
CLIENT_ID	NUMBER(5)			

SQL> desc Include

Name	Type	Nullable	Default	Comments
PROFESSIONAL_ID	NUMBER(5)			
ORDER_ID	NUMBER(5)			

SQL> desc Payment

Name	Type	Nullable	Default	Comments
PAYMENT_ID	NUMBER(5)			
PAYMENT_AMOUNT	FLOAT			
PAYMENT_DATE	DATE			
PAY_METHOD_ID	NUMBER(5)			
ORDER_ID	NUMBER(5)			

שאלות

Select שאלות

שאלה 1:

השאלה תמצא את כל הלקוחות שהם בעלי יותר משני אירועים שונים, תחשב את הסכום הכולל של התשלומים שהם שילמו, ותציין את החודש של התאריך האחרון שבו השתתפו באירוע.

```
SELECT
    c.client_id,
    c.fname AS client_fname,
    c.lname AS client_lname,
    event_data.total_events,
    payment_data.total_payments,
    EXTRACT(MONTH FROM event_data.last_event_date) AS last_event_month
FROM
    Client c
JOIN
    (SELECT
        co.client_id,
        COUNT(DISTINCT co.event_id) AS total_events,
        MAX(e.event_date) AS last_event_date
    FROM
        CustomerOrder co
    JOIN
        Event e ON co.event_id = e.event_id
    GROUP BY
        co.client_id
    HAVING
        COUNT(DISTINCT co.event_id) > 2) event_data
ON
    c.client_id = event_data.client_id
JOIN
    (SELECT
        co.client_id,
        SUM(p.payment_amount) AS total_payments
    FROM
        CustomerOrder co
    JOIN
        Payment p ON co.order_id = p.order_id
    GROUP BY
        co.client_id) payment_data
ON
    c.client_id = payment_data.client_id
ORDER BY
    event_data.total_events DESC;
```

	CLIENT_ID	CLIENT_FNAME	CLIENT_LNAME	TOTAL_EVENTS	TOTAL_PAYMENTS	LAST_EVENT_MONTH
1	135	Chris	Vega	6	187500	11
2	154	Domingo	MacPherson	4	84000	3
3	230	Ryan	Richter	4	106000	11
4	187	Drew	Cassel	4	107200	4
5	299	Shannyn	Cara	4	130000	2

שאלתה 2:

השאלתה תמצא את כל אנשי המקצוע שהוזמנו ליותר מ-3 אירועים שונים (שייכים ל 3 הזמנות שונות), תחשב את הסכום הכולל של התשלומים שהם קיבלו.

```

SELECT
  p.professional_id,
  p.fname AS professional_fname,
  p.lname AS professional_lname,
  event_data.total_orders,
  payment_data.total_payments
FROM
  Professional p
JOIN
  (SELECT
    i.professional_id,
    COUNT(DISTINCT co.order_id) AS total_orders
  FROM
    include i
  JOIN
    CustomerOrder co ON i.order_id = co.order_id
  GROUP BY
    i.professional_id
  HAVING
    COUNT(DISTINCT co.order_id) > 3) event_data
ON
  p.professional_id = event_data.professional_id
JOIN
  (SELECT
    i.professional_id,
    SUM(p.payment_amount) AS total_payments
  FROM
    include i
  JOIN
    Payment p ON i.order_id = p.order_id
  GROUP BY
    i.professional_id) payment_data
ON
  p.professional_id = payment_data.professional_id
ORDER BY
  p.professional_id

```

	PROFESSIONAL_ID	PROFESSIONAL_FNAME	PROFESSIONAL_LNAME	TOTAL_ORDERS	TOTAL_PAYMENTS
1	388	Anna	Garcia	4	141000
2	295	John	Hernandez	4	132900
3	64	Mike	Garcia	4	114000
4	175	Paul	Hernandez	4	110400
5	151	Alex	Hernandez	5	109000
6	209	Mike	Davis	4	98000
7	59	Jane	Miller	4	95000
8	347	Chris	Miller	4	89200

שאלתה 3:

השאלתה תמצא את האירוע האחרון לכל לקוח כולל פרטיו.

```
SELECT
    c.client_id,
    c.fname AS client_fname,
    c.lname AS client_lname,
    e.event_id,
    e.event_date,
    e.place
FROM
    Client c,
    CustomerOrder co,
    Event e
WHERE
    c.client_id = co.client_id
    AND co.event_id = e.event_id
    AND e.event_date = (
        SELECT
            MAX(e2.event_date)
        FROM
            Event e2,
            CustomerOrder co2
        WHERE
            e2.event_id = co2.event_id
            AND co2.client_id = c.client_id
    );
```

	CLIENT_ID	CLIENT_FNAME	CLIENT_LNAME	EVENT_ID	EVENT_DATE	PLACE
1	0	Arnold	Hornsby	88	25/02/2020	Grand Hall
2	1	Bruce	Baranski	303	11/04/2015	Exhibition Hall
3	3	Bryan	Snipes	372	26/12/2017	Grand Hall
4	4	Rip	Palmer	382	10/06/2010	Grand Hall
5	6	Luis	Saucedo	241	24/04/2023	Event Arena
6	9	Patricia	Tolkan	121	16/07/2014	Conference Center
7	10	Eliza	Reeve	340	15/04/2013	Banquet Hall
8	11	Caroline	Black	27	25/08/2000	Conference Center

שאלתה 4:

השאלתה תמצא את הלקוחות שביצעו הזמנות ביותר משיטת תשלום אחת ותציג את מספר שיטות התשלום לכל אחד מהם.

```
SELECT
    c.client_id,
    c.fname AS client_fname,
    c.lname AS client_lname,
    COUNT(DISTINCT p.pay_method_id) AS unique_payment_methods
FROM
    Client c,
    CustomerOrder co,
    Payment p
WHERE
    c.client_id = co.client_id
    AND co.order_id = p.order_id
GROUP BY
    c.client_id, c.fname, c.lname
HAVING
    COUNT(DISTINCT p.pay_method_id) > 1
ORDER BY
    unique_payment_methods DESC;
```

		CLIENT_ID	CLIENT_FNAME	CLIENT_LNAME	UNIQUE_PAYMENT_METHODS
▶	1	135	Chris	Vega	4
	2	145	Larry	Elizondo	4
	3	161	Winona	Reeves	3
	4	211	Tony	Graham	3
	5	72	Ernest	Singh	3
	6	183	Norm	Ryder	3
	7	55	Kenny	Allison	3

שאלות Update

שאלת 1:

תיאור: השאלתה מעדכנת את טבלת ההזמנות בשורות בהן מתקיים שתאריך הזמנה גדול או שווה לתאריך האירוע שלה. תאריך ההזמנה משתנה להיות יום לפני תאריך האירוע.

מצב לפני הפעלת השאלתה:

SQL			
Output			
Statistics			
<pre>select * from Customerorder; select * from Event;</pre>			
Select customerorder			
Select event			
EVENT_ID	EVENT_DATE	PLACE	
178	05/09/2011	Grand Hall	...
179	26/11/2007	Event Arena	...
180	14/05/2016	Banquet Hall	...
181	11/12/2005	Grand Hall	...
182	21/07/2009	Banquet Hall	...

SQL			
Output			
Statistics			
<pre>select * from Customerorder; select * from Event;</pre>			
Select customerorder			
Select event			
ORDER_ID	ORDER_DATE	EVENT_ID	CLIENT_ID
1	289 11/10/2022	177	59
2	290 06/01/2020	224	26
3	291 22/04/2024	343	220
4	292 03/01/2020	314	205
5	293 25/03/2020	383	68
6	294 20/08/2020	121	204
7	295 27/09/2022	21	351
8	296 10/10/2022	160	160

מצב אחרי הפעלת השאלתה:

UPDATE CustomerOrder			
SET order_date = (
SELECT event_date - 1			
FROM Event			
WHERE Event.event_id = CustomerOrder.event_id			
)			
WHERE order_date >= (
SELECT event_date			
FROM Event			
WHERE Event.event_id = CustomerOrder.event_id			
);			
Select * from Customerorder;			
select * from Event;			
Select customerorder			
Select event			
ORDER_ID	ORDER_DATE	EVENT_ID	CLIENT_ID
1	289 04/09/2011	177	59
2	290 06/01/2020	224	26
3	291 11/01/2010	343	220
4	292 22/02/2001	314	205
5	293 05/07/2015	383	68
6	294 15/07/2014	121	204

שאלת 2:

תיאור: השאלתה מעדכנת את טבלת התשלומים בשורות בהן מתקיים שתאריך הזמנה קטן מתאריך התשלום שלה. (להזמנה יכולים להיות כמה תשלומים)

השאלתה מעדכנת בשורות אלו את תאריך התשלום להיות שווה לתאריך ההזמנה.

```
select * from PAYMENT
select * from CUSTOMERORDER
```

מצב לפני הפעלת השאילתה:

	ORDER_ID	ORDER_DATE	EVENT_ID	CLIENT_ID
20	7	02/12/2021	8	283
21	8	15/07/2020	13	329
22	9	22/04/2021	130	92
23	10	28/08/2020	346	29
24	11	23/05/2022	296	277
25	12	18/05/2021	79	349
26	13	01/03/2023	162	385
27	14	12/02/2020	54	379
28	15	05/04/2024	132	315
29	16	04/10/2023	329	238
30	17	23/01/2022	257	251
31	18	25/01/2024	97	13
32	19	16/08/2023	108	135
33	20	30/08/2021	360	168
34	21	01/07/2021	204	135
35	22	18/02/2021	256	386
36	23	06/01/2022	128	161
37	24	26/04/2023	381	381
38	25	07/05/2022	43	102
39	26	18/08/2022	144	333

	PAYMENT_ID	PAYMENT_AMOUNT	PAYMENT_DATE	PAY_METHOD_ID	ORDER_ID
13	12	14000	28/05/2020	0	12
14	13	24000	19/02/2023	3	13
15	14	37000	06/12/2020	0	14
16	15	36000	18/04/2020	2	15
17	16	31000	13/05/2023	0	16
18	17	37000	21/02/2021	2	17
19	18	30000	21/09/2021	2	18
20	19	30000	25/02/2020	2	19
21	20	29000	01/10/2020	2	20
22	21	29000	26/07/2023	0	21
23	22	33000	20/10/2020	3	22
24	23	38000	19/01/2020	3	23
25	24	48000	15/08/2022	3	24
26	25	49000	07/04/2020	3	25

מצב אחרי הפעלת השאילתה:

SQL

Output

Statistics

```

UPDATE Payment
SET payment_date = (
  SELECT order_date
  FROM CustomerOrder
  WHERE CustomerOrder.order_id = Payment.order_id
)
WHERE payment_date < (
  SELECT order_date
  FROM CustomerOrder
  WHERE CustomerOrder.order_id = Payment.order_id
);

select * from Customerorder;
select * from Payment;

```

Select customerorder

Select payment

	PAYMENT_ID	PAYMENT_AMOUNT	PAYMENT_DATE	PAY_METHOD_ID	ORDER_ID
23	22	33000	20/10/2020	3	22
24	23	38000	06/01/2022	3	23
25	24	48000	15/08/2022	3	24
26	25	49000	07/04/2020	3	25
27	26	6000	13/10/2023	3	26

שאילתה 3:

תיאור: השאילתה מעדכנת את טבלת ההזמנות כך שמספר ההזמנה יהיה זהה למספר האירוע. זאת כדי לוודא את הקשר של 1 ל 1, שלכל אירוע יש הזמנה יחידה.

```

UPDATE CustomerOrder
SET event_id=order_id

select * from Customerorder

```

	ORDER_ID	ORDER_DATE	EVENT_ID	CLIENT_ID
1	289	11/10/2022	289	59
2	290	06/01/2020	290	26
3	291	22/04/2024	291	220
4	292	03/01/2020	292	205
5	293	25/03/2020	293	68
6	294	20/08/2020	294	204
7	295	27/09/2022	295	351
8	296	23/03/2023	296	230
9	297	18/04/2022	297	27
10	298	11/01/2021	298	109
11	299	16/03/2020	299	105
12	300	26/03/2023	300	362

שאלות Delete

שאלה 1:

תיאור: השאלה מוחקת את ההזמנות שנוצרו לפני שנת 2001 (ואת כל המידע בטבלאות השונות שקשור אליהן)

מסד הנתונים לפני המחיקה:

```
select * from CUSTOMERORDER WHERE EXTRACT (YEAR FROM ORDER_DATE) < 2001
```

	ORDER_ID	ORDER_DATE	EVENT_ID	CLIENT_ID
1	289	06/10/2000	289	59
2	4	13/04/2000	4	341
3	26	17/01/2000	26	333
4	27	24/08/2000	27	11
5	39	15/04/2000	39	201
6	79	12/01/2000	79	112
7	86	22/03/2000	86	350
8	134	06/03/2000	134	373
9	136	27/06/2000	136	108
10	187	20/06/2000	187	103
11	258	06/03/2000	258	215
12	266	08/08/2000	266	215

השאלות המורצות:

```
-- Delete payments for the orders older than 30 years
DELETE FROM Payment
WHERE order_id IN (
    SELECT order_id FROM CustomerOrder
    WHERE EXTRACT (YEAR FROM ORDER_DATE) < 2001
);

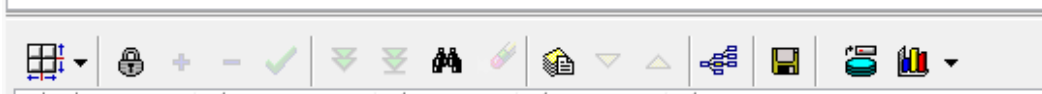
-- Delete entries from the include table for the orders older than 30 years
DELETE FROM include
WHERE order_id IN (
    SELECT order_id FROM CustomerOrder
    WHERE EXTRACT (YEAR FROM ORDER_DATE) < 2001
);

-- Delete the orders older than 30 years
DELETE FROM CustomerOrder
WHERE EXTRACT (YEAR FROM ORDER_DATE) < 2001;

-- Delete events associated with orders older than 30 years
DELETE FROM Event
WHERE event_id IN (
    SELECT event_id FROM CustomerOrder
    WHERE EXTRACT (YEAR FROM ORDER_DATE) < 2001
);
```


מסד הנתונים אחרי המחיקה:

```
select * from event WHERE EXTRACT(YEAR FROM ORDER_DATE) < 2001
```




	ORDER_ID	ORDER_DATE	EVENT_ID	CLIENT_ID
--	----------	------------	----------	-----------

שאלת 2:

תיאור: השאלתה מוחקת את בעלי המקצוע שלא הוזמנו משנת 2001. (ואת כל המידע בטבלאות השונות שקשור אליהן)

מסד הנתונים לפני המחיקה:

```
-- Delete professionals who haven't taken any orders from 2001
SELECT * FROM Professional
WHERE professional_id NOT IN (
  SELECT professional_id
  FROM include
  JOIN CustomerOrder
  ON include.order_id = CustomerOrder.order_id
  WHERE EXTRACT(YEAR FROM CustomerOrder.order_date) > 2001
);
```



	PROFESSIONAL_ID	FNAME	LNAME	PHONE	EMAIL	PRICE	START_YEAR	PROF_ID
1	3	Katie	Miller	5573072267	katie.miller@gmail.com	41400	2005	1
2	4	Sara	Hernandez	5506837826	sara.hernandez@gmail.com	25700	2009	0
3	6	Anna	Garcia	5578516792	anna.garcia@gmail.com	41200	2022	0
4	8	Katie	Smith	5571446153	katie.smith@gmail.com	12200	2018	0
5	9	Alex	Williams	5525793180	alex.williams@gmail.com	33100	2017	0
6	12	Jane	Hernandez	5597242297	jane.hernandez@gmail.com	15200	2010	0
7	13	John	Jones	5504885227	john.jones@gmail.com	35200	2004	1
8	20	Alex	Martinez	5561173787	alex.martinez@gmail.com	17200	2018	1
9	23	Anna	Garcia	5505771937	anna.garcia@gmail.com	39500	2007	1
10	24	Alex	Jones	5507451824	alex.jones@gmail.com	48100	2017	0
11	25	Sara	Davis	5512744715	sara.davis@gmail.com	32300	2023	1
12	27	Jane	Smith	5534052519	jane.smith@gmail.com	40200	2014	0
13	28	Tom	Miller	5504789135	tom.miller@gmail.com	32000	2022	1

השאלות המורצות:


```
SQL Output Statistics

-- Delete from the include table where the professional hasn't taken any orders from 2001
DELETE FROM include
WHERE professional_id IN (
  SELECT professional_id
  FROM Professional
  WHERE professional_id NOT IN (
    SELECT professional_id
    FROM include
    JOIN CustomerOrder ON include.order_id = CustomerOrder.order_id
    WHERE EXTRACT(YEAR FROM CustomerOrder.order_date) > 2001
  )
);

-- Delete professionals who haven't taken any orders from 2001
delete FROM Professional
WHERE professional_id NOT IN (
  SELECT professional_id
  FROM include
  JOIN CustomerOrder
  ON include.order_id = CustomerOrder.order_id
  WHERE EXTRACT(YEAR FROM CustomerOrder.order_date) > 2001
);
```

מסד הנתונים אחרי המחיקה:

```
-- Delete professionals who haven't taken any orders from 2001
SELECT * FROM Professional
WHERE professional_id NOT IN (
  SELECT professional_id
  FROM include
  JOIN CustomerOrder
  ON include.order_id = CustomerOrder.order_id
  WHERE EXTRACT(YEAR FROM CustomerOrder.order_date) > 2001
);
```



	PROFESSIONAL_ID	FNAME	LNAME	PHONE	EMAIL	PRICE	START_YEAR	PROF_ID
--	-----------------	-------	-------	-------	-------	-------	------------	---------

אילוצים

אילוץ 1:

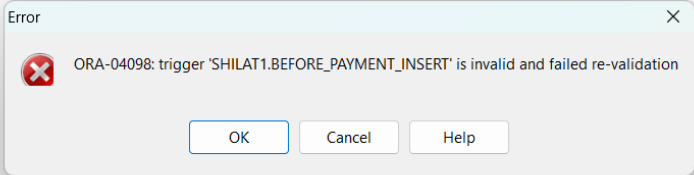
תיאור: השאילתה מאלצת בעת הכנסת תשלום למסד הנתונים שתאריך התשלום יהיה גדול שווה לתאריך ההזמנה.

```
BEFORE INSERT ON Payment
FOR EACH ROW
BEGIN
    DECLARE order_date INT;

    -- Fetch the order_date for the new payment
    SELECT order_date INTO order_date
    FROM CustomerOrder
    WHERE CustomerOrder.order_id = NEW.order_id;

    -- Check if the new payment_date is after the order_date
    IF NEW.payment_date < order_date THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Payment date must be after order date';
    END IF;
END

-- This should fail because the payment_date is before the order_date
INSERT INTO Payment (payment_id, payment_amount, payment_date, pay_method_id, order_id) VALUES (400, 20000, 9/10/2022, 2, 289 );
```



אילוץ 2:

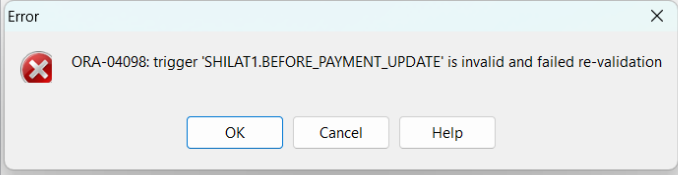
תיאור: השאילתה מאלצת בעת עדכון תשלום במסד הנתונים שתאריך התשלום יהיה גדול שווה לתאריך ההזמנה.

```
-- Trigger for UPDATE operations on Payment table
CREATE TRIGGER before_payment_update
BEFORE UPDATE ON Payment
FOR EACH ROW
BEGIN
    DECLARE order_date INT; -- Declare a local variable to hold the order_date

    -- Fetch the order_date for the updated payment's order_id
    SELECT order_date INTO order_date
    FROM CustomerOrder
    WHERE CustomerOrder.order_id = NEW.order_id;

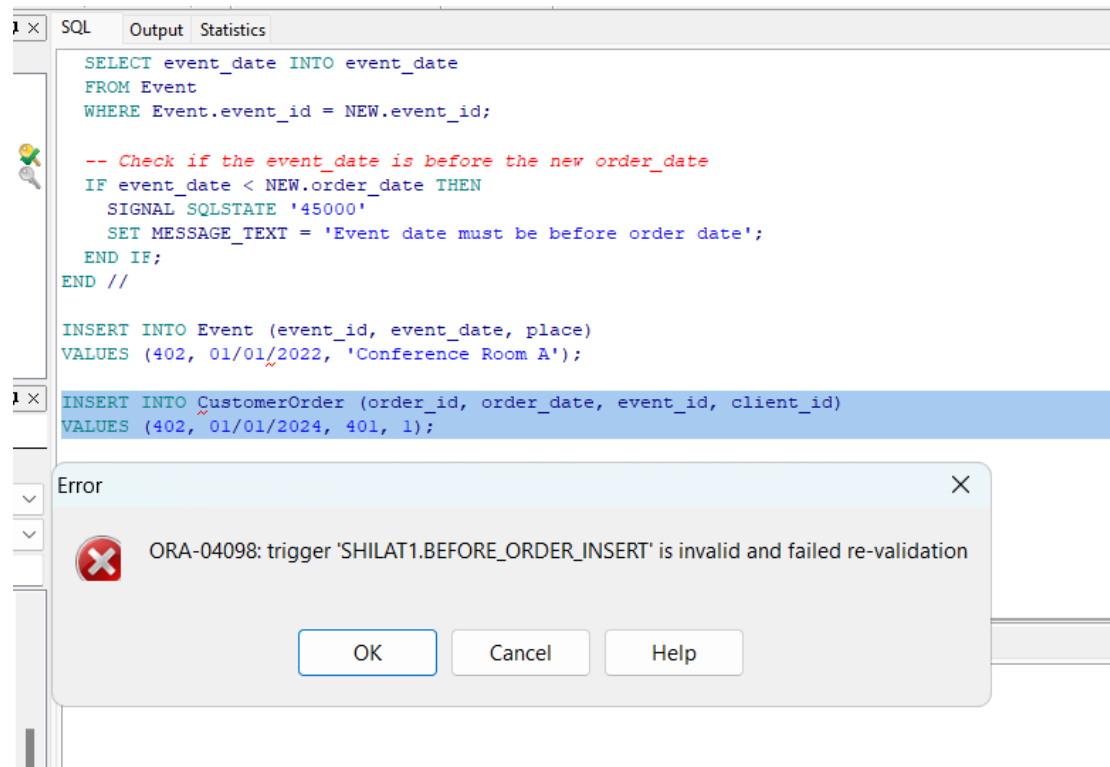
    -- Check if the updated payment_date is after the order_date
    IF NEW.payment_date < order_date THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Payment date must be after order date'; -- Raise an error if condition is not met
    END IF;
END //

-- This update should fail because the updated payment_date is before the order_date
UPDATE Payment
SET payment_date = 10/10/2022
WHERE payment_id = 289;
```



אילון 3:

תיאור: השאילתה מאלצת בעת הכנסת הזמנה למסד הנתונים שתאריך ההזמנה יהיה קטן מתאריך האירוע המשותף אליה.



The screenshot shows the SQL Developer interface with a SQL window containing the following code:

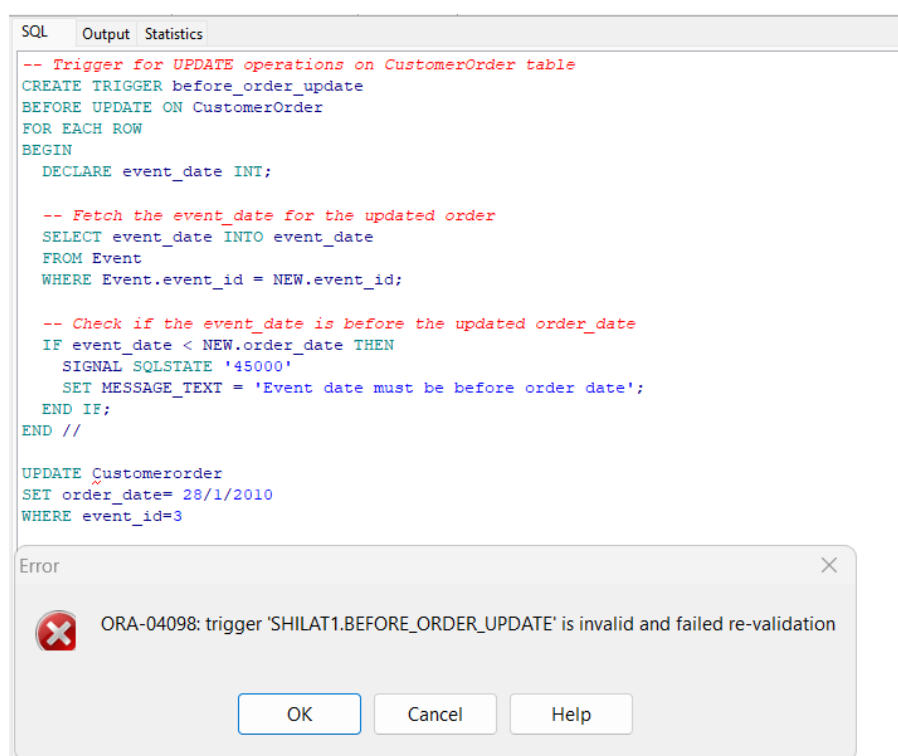
```
SELECT event_date INTO event_date
FROM Event
WHERE Event.event_id = NEW.event_id;

-- Check if the event_date is before the new order_date
IF event_date < NEW.order_date THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Event date must be before order date';
END IF;
END //
```

Below the code, there is an `INSERT INTO CustomerOrder` statement. An error dialog box is displayed with the message: "ORA-04098: trigger 'SHILAT1.BEFORE_ORDER_INSERT' is invalid and failed re-validation". The dialog has OK, Cancel, and Help buttons.

אילון 4:

תיאור: השאילתה מאלצת בעת עדכון הזמנה במסד הנתונים שתאריך ההזמנה יהיה קטן מתאריך האירוע המשותף אליה.



The screenshot shows the SQL Developer interface with a SQL window containing the following code:

```
-- Trigger for UPDATE operations on CustomerOrder table
CREATE TRIGGER before_order_update
BEFORE UPDATE ON CustomerOrder
FOR EACH ROW
BEGIN
    DECLARE event_date INT;

    -- Fetch the event_date for the updated order
    SELECT event_date INTO event_date
    FROM Event
    WHERE Event.event_id = NEW.event_id;

    -- Check if the event_date is before the updated order_date
    IF event_date < NEW.order_date THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Event date must be before order date';
    END IF;
END //
```

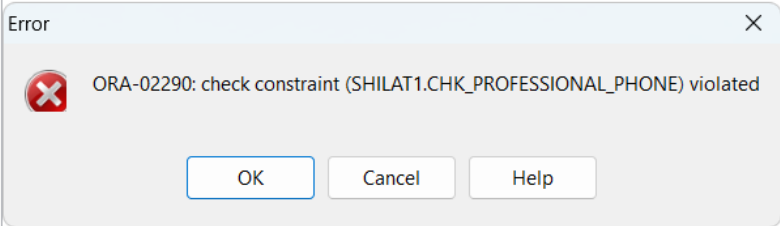
Below the code, there is an `UPDATE CustomerOrder` statement. An error dialog box is displayed with the message: "ORA-04098: trigger 'SHILAT1.BEFORE_ORDER_UPDATE' is invalid and failed re-validation". The dialog has OK, Cancel, and Help buttons.

אילוז 5:

תיאור: השאילתה מאלצת את מספר הטלפון של בעל מקצוע להתחיל ב 5.

```
ALTER TABLE Professional
ADD CONSTRAINT chk_professional_phone
CHECK (phone LIKE '5%');

INSERT INTO Professional (professional_id, fname, lname, phone, email, price, start_year, prof_id)
VALUES (401, 'John', 'Doe', '123456789', 'john.doe@example.com', 100, 2020, 1);
```



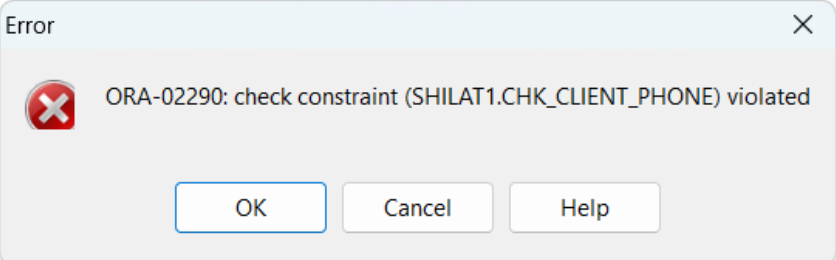
An error dialog box titled "Error" with a red 'X' icon. The message reads: "ORA-02290: check constraint (SHILAT1.CHK_PROFESSIONAL_PHONE) violated". At the bottom are three buttons: "OK", "Cancel", and "Help".

אילוז 6:

תיאור: השאילתה מאלצת את מספר הטלפון של לקוח להתחיל ב 5.

```
-- Add a CHECK constraint to the Client table
ALTER TABLE Client
ADD CONSTRAINT chk_client_phone
CHECK (phone LIKE '5%');

INSERT INTO Client (client_id, fname, lname, address, phone, email)
VALUES (402, 'John', 'Doe', '123 Main St', '0001234567', 'john.doe@example.com');
```



An error dialog box titled "Error" with a red 'X' icon. The message reads: "ORA-02290: check constraint (SHILAT1.CHK_CLIENT_PHONE) violated". At the bottom are three buttons: "OK", "Cancel", and "Help".

שאלות עם פרמטרים

שאלת 1:

תיאור: מחזירה את כל הנתונים הרלוונטיים על בעלי מקצוע לאחר סינון לפי תנאים דינמיים:

בעלי מקצוע שהתחילו לעבוד אחרי שנה נתונה ושגובים מחיר גבוה יותר ממחיר נתון.

```
SELECT
  p.professional_id,
  p.fname,
  p.lname,
  p.phone,
  p.email,
  p.price,
  p.start_year,
  pr.prof_name
FROM
  Professional p
JOIN
  Profession pr ON p.prof_id = pr.prof_id
WHERE
  p.start_year > <name="start_year" hint="a number between 1980 to 2024" type="integer" required="true">
  AND p.price > <name="min_price" hint="a number between 1000 to 100000" type="integer" required="true">;
```

	PROFESSIONAL_ID	FNAME	LNAME	PHONE	EMAIL	PRICE	START_YEAR	PROF_NAME
1	0	John	Smith	5511652243	john.smith@gmail.com	37700	2020	photographer
2	6	Anna	Garcia	5578516792	anna.garcia@gmail.com	41200	2022	photographer
3	16	Paul	Garcia	5580465762	paul.garcia@gmail.com	39600	2022	video
4	25	Sara	Davis	5512744715	sara.davis@gmail.com	32300	2023	video
5	26	Alex	Martinez	5584255342	alex.martinez@gmail.com	20900	2020	photographer
6	28	Tom	Miller	5504789135	tom.miller@gmail.com	32000	2022	video
7	33	Sara	Johnson	5568263489	sara@gmail.com	44000	2023	video
8	35	Mike	Garcia	5564252363	mike@gmail.com	32000	2020	video
9	43	Sara	Miller	5571062693	sara@gmail.com	46000	2023	photographer
10	59	Jane	Miller	5568960162	jane@gmail.com	20000	2020	photographer
11	60	Jane	Brown	5588209974	jane@gmail.com	23000	2020	video
12	78	Katie	Miller	5580555878	katie@gmail.com	28000	2021	video
13	86	Paul	Johnson	5548935464	paul@gmail.com	25000	2021	video
14	88	Paul	Jones	5559135978	paul@gmail.com	34000	2022	photographer
15	104	Anna	Hernandez	5524104193	anna@gmail.com	37000	2022	photographer
16	107	Mike	Johnson	5599793118	mike@gmail.com	35000	2023	video
17	110	Mike	Brown	5563941169	mike@gmail.com	44000	2023	video
18	123	Jane	Williams	5516819652	jane@gmail.com	27000	2020	video
19	147	Alex	Johnson	5505603869	alex@gmail.com	47000	2021	video

Variables

Name	Value
start_year	2019
min_price	18000

OK Cancel Clear

a number between 1980 to 2024

שאלת 2:

תיאור: השאלת תחזיר את פרטי האירועים יחד עם פרטי הלקוחות המשתתפים בהם, כאשר המקומות של האירועים יהיו באולם מתוך הרשימה שנבחרה (הרשימה המוגדרת בפרמטר)

```
SELECT
  e.event_id,
  e.event_date,
  e.place,
  c.client_id,
  c.fname AS client_fname,
  c.lname AS client_lname
FROM
  Event e
JOIN
  CustomerOrder co ON e.event_id = co.event_id
JOIN
  Client c ON co.client_id = c.client_id
WHERE
  e.place=<name="place_list" type="string" list="select place from event" restricted="no">;
```

Variables

Name	Value
place_list	Grand Hall

OK Cancel Clear

	EVENT_ID	EVENT_DATE	PLACE	CLIENT_ID	CLIENT_FNAME	CLIENT_LNAME
1	88	25/02/2020	Grand Hall	0	Arnold	Hornby
2	372	26/12/2017	Grand Hall	3	Bryan	Snipes
3	382	10/06/2010	Grand Hall	4	Rip	Palmer
4	259	20/04/2009	Grand Hall	10	Eliza	Reeve
5	152	12/07/2013	Grand Hall	12	Dave	Cotton
6	18	01/06/2004	Grand Hall	13	Cheryl	Gibbons
7	99	01/11/2001	Grand Hall	21	Annette	De Niro
8	69	27/01/2013	Grand Hall	23	Grant	Peebles
9	150	29/09/2001	Grand Hall	24	Frances	Fisher
10	10	03/05/2006	Grand Hall	29	Corey	Reed
11	214	06/04/2024	Grand Hall	34	Mili	Sample
12	271	02/03/2014	Grand Hall	35	Nina	DeLuiise
13	169	02/04/2008	Grand Hall	36	Xander	Baker
14	130	01/01/2010	Grand Hall	39	Elizabeth	McNarland

שאלת 3:

תיאור: השאלתה מציגה פרטי אירועים, סוגי התשלומים שבוצעו עבורם וסכום התשלום, תוך שימוש בפרמטרים דינמיים לסינון לפי טווח תאריכים ומקום האירוע. השאלתה ממיינת את התוצאות לפי תאריך האירוע.

The screenshot shows a SQL IDE window titled "SQL Window - SELECT e.event_id, e.event_date, e.place, pm.type AS payment_type, p.payment_amount FROM Event e ...". The query is as follows:

```
SELECT
    e.event_id,
    e.event_date,
    e.place,
    pm.type AS payment_type,
    p.payment_amount
FROM
    Event e
JOIN
    CustomerOrder co ON e.event_id = co.event_id
JOIN
    Payment p ON co.order_id = p.order_id
JOIN
    PaymentMethod pm ON p.pay_method_id = pm.pay_method_id
WHERE
    e.event_date BETWEEN TO_DATE(&<name="start_date" type="string" required="true" hint="Format: dd/mm/yyyy ">, 'dd/mm/yyyy')
    AND TO_DATE(&<name="end_date" type="string" required="true" hint="Format: dd/mm/yyyy ">, 'dd/mm/yyyy')
    AND e.place IN (&<name="place_list" type="string" list="select distinct place from event" restricted="no">)
ORDER BY
    e.event_date;
```

The results are displayed in a table with the following columns: EVENT_ID, EVENT_DATE, PLACE, PAYMENT_TYPE, and PAYMENT_AMOUNT. The table contains 17 rows of data. A "Variables" dialog box is open, showing the following values:

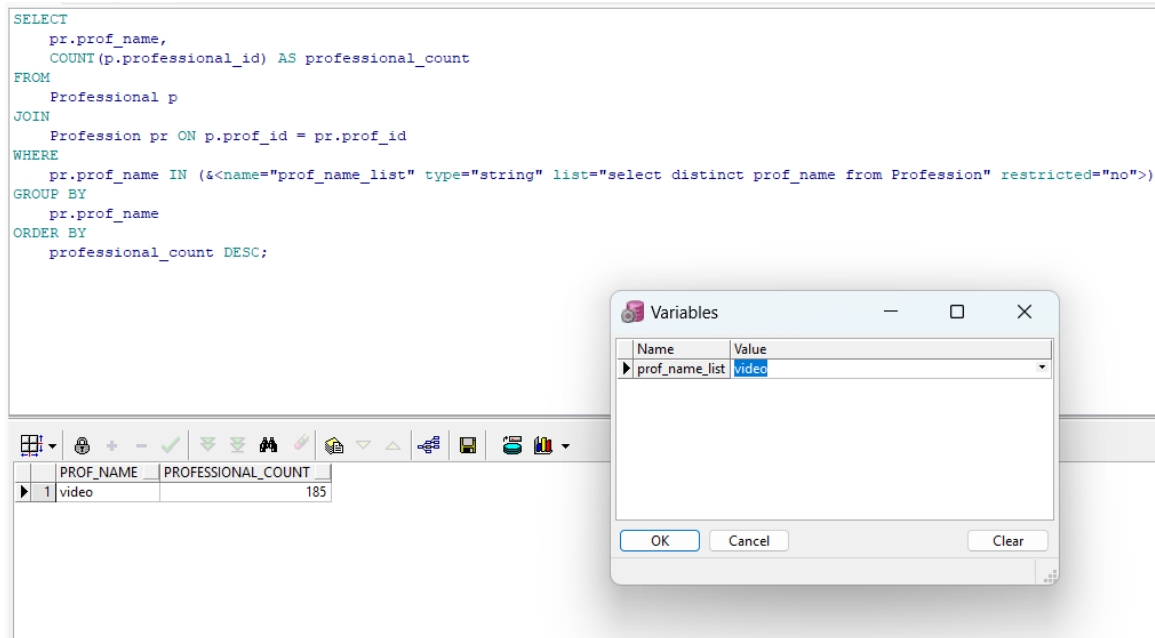
Name	Value
start_date	1/1/2015
end_date	2/2/2024
place_list	Event Arena

The "Variables" dialog box also has "OK", "Cancel", and "Clear" buttons, and a "Format: dd/mm/yyyy" label.

שאלתה 4:

תיאור: שאלתה המאפשרת לבחור ממקצועות זמינים בטבלת בעלי המקצוע ומציגה את מספרם.

```
SELECT
    pr.prof_name,
    COUNT(p.professional_id) AS professional_count
FROM
    Professional p
JOIN
    Profession pr ON p.prof_id = pr.prof_id
WHERE
    pr.prof_name IN (&<name="prof_name_list" type="string" list="select distinct prof_name from Profession" restricted="no">)
GROUP BY
    pr.prof_name
ORDER BY
    professional_count DESC;
```



	PROF_NAME	PROFESSIONAL_COUNT
1	video	185

פונקציות

פונקציה 1:

תיאור: הפונקציה מקבלת מספר מזהה של בעל מקצוע ותאריך. הפונקציה בודקת האם הבעל מקצוע פנוי בתאריך הזה ומחזירה ערך בוליאני בהתאמה. בנוסף, הפונקציה מדפיסה את פרטי האירועים שמונעים מבעל המקצוע להיות פנוי בתאריך המדובר, אם יש כאלו.

```
CREATE OR REPLACE FUNCTION check_p_availability(p_id IN INT, p_date IN DATE)
RETURN BOOLEAN IS
    CURSOR prof_events IS
        SELECT e.event_id, e.event_date, e.place, co.order_id
        FROM CustomerOrder co
        JOIN include i ON co.order_id = i.order_id
        JOIN Event e ON co.event_id = e.event_id
        WHERE i.professional_id = p_id
        AND e.event_date = p_date;
    event_count NUMBER := 0;
    prof_event_row prof_events%ROWTYPE;
    is_available BOOLEAN;
BEGIN
    FOR prof_event_row IN prof_events LOOP
        event_count := event_count + 1;

        IF event_count = 1 THEN
            DBMS_OUTPUT.PUT_LINE('Professional is not available on ' || TO_CHAR(p_date, 'YYYY-MM-DD') || '. Event details:');
        END IF;

        DBMS_OUTPUT.PUT_LINE('Event ID: ' || prof_event_row.event_id);
        DBMS_OUTPUT.PUT_LINE('Event Date: ' || TO_CHAR(prof_event_row.event_date, 'YYYY-MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Place: ' || prof_event_row.place);
        DBMS_OUTPUT.PUT_LINE('Order ID: ' || prof_event_row.order_id);
        DBMS_OUTPUT.PUT_LINE('---');
    END LOOP;

    is_available := (event_count = 0);

    IF is_available THEN
        DBMS_OUTPUT.PUT_LINE('Professional is available on ' || TO_CHAR(p_date, 'YYYY-MM-DD'));
    ELSE
        DBMS_OUTPUT.PUT_LINE('Total events found: ' || event_count);
    END IF;

    RETURN is_available;
END check_p_availability;
```

דוגמת הרצה:

```
1 declare
2     -- Boolean parameters are translated from/to integers:
3     -- 0/1/null <--> false/true/null
4     result boolean;
5 begin
6     -- Call the function
7     result := check_p_availability(p_id => :p_id,
8                                   p_date => :p_date);
9     -- Convert false/true/null to 0/1/null
10    :result := sys.diutil.bool_to_int(result);
11 end;
```

<input type="checkbox"/>	Variable	Type	Value
<input checked="" type="checkbox"/>	result	Integer	0
<input checked="" type="checkbox"/>	p_id	Float	5
<input checked="" type="checkbox"/>	p_date	Date	25/09/2022
<input checked="" type="checkbox"/>	*		

Test script DBMS Output Statistics Profiler Trace

Clear Buffer size 10000 ☒ Enabled

```

Professional is not available on 2022-09-25. Event details:
Event ID: 337
Event Date: 2022-09-25
Place: Banquet Hall
Order ID: 337
---
Total events found: 1

```

הוכחה מבסיס הנתונים:

```

SELECT *
FROM CustomerOrder co
JOIN include i ON co.order_id = i.order_id
JOIN Event e ON co.event_id = e.event_id
where i.professional_id =5

```

Select customerorder		Select customerorder									
	ORDER_ID	ORDER_DATE	EVENT_ID	CLIENT_ID	PROFESSIONAL_ID	ORDER_ID	EVENT_ID	EVENT_DATE	PLACE		
1	337	30/06/2022	337	88	5	337	337	25/09/2022	Banquet Hall	...	

פונקציה 2:

תיאור: הפונקציה מקבלת מספר מזהה של הזמנה. הפונקציה תחזיר את הסכום שהלקוח חייב על ההזמנה. היא תחשב את המחיר הכולל של ההזמנה ואת הסכום הכולל ששולם. אם שולם יותר מהנדרש, הפונקציה תזרוק חריגה עם הסכום להחזר. אם יש עוד לתשלום, הפונקציה תחזיר את הסכום שנותר לשלם.

```
1 CREATE OR REPLACE FUNCTION calculate_remaining_payment(p_order_id IN NUMBER)
2 RETURN NUMBER IS
3     v_total_price NUMBER := 0;
4     v_total_paid NUMBER := 0;
5     v_remaining NUMBER := 0;
6
7     CURSOR c_professionals IS
8         SELECT p.price
9         FROM Professional p
10        JOIN include i ON p.professional_id = i.professional_id
11        WHERE i.order_id = p_order_id;
12
13     CURSOR c_payments IS
14         SELECT payment_amount
15         FROM Payment
16        WHERE order_id = p_order_id;
17
18 BEGIN
19     -- Calculate total price
20     FOR prof_rec IN c_professionals LOOP
21         v_total_price := v_total_price + prof_rec.price;
22     END LOOP;
23
24     -- Calculate total paid
25     FOR payment_rec IN c_payments LOOP
26         v_total_paid := v_total_paid + payment_rec.payment_amount;
27     END LOOP;
28
29     -- Calculate remaining payment
30     v_remaining := v_total_price - v_total_paid;
31
32     -- Raise exception if overpaid
33     IF v_remaining < 0 THEN
34         RAISE_APPLICATION_ERROR(-20001, 'Overpayment detected. Refund amount: ' || ABS(v_remaining));
35     END IF;
36
37     RETURN v_remaining;
38 EXCEPTION
39     WHEN OTHERS THEN
40         DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
41         RETURN NULL;
42 END calculate_remaining_payment;
```

דוגמת הרצה:

```
1 begin
2   -- Call the function
3   :result := calculate_remaining_payment(p_order_id => :p_order_id);
4 end;
```

<input type="checkbox"/>	Variable	Type	Value
<input checked="" type="checkbox"/>	result	Float	21000
<input checked="" type="checkbox"/>	p_order_id	Float	5
<input checked="" type="checkbox"/>	*		

דוגמת הרצה עם זריקת חריגה:

<input type="checkbox"/>	Variable	Type	Value
<input checked="" type="checkbox"/>	result	Float	
<input checked="" type="checkbox"/>	p_order_id	Float	88
<input checked="" type="checkbox"/>	*		

SQL*Plus: Release 11.1.0.7.0 - Production on Mon Jun 11 11:11:11 2013
Copyright (c) 2006 Oracle Corporation
All rights reserved.

An error occurred: ORA-20001: Overpayment detected. Refund amount: 14000

פרוצדורות

פרוצדורה 1:

תיאור: הפרוצדורה מקבלת מספר מזהה של בעל מקצוע ותאריך. הפרוצדורה מדפיסה את המחליפים שיכולים להיות לבעל המקצוע הזה באותו התאריך. אם אין כאלו, תודפס הודעה מתאימה.

```
CREATE OR REPLACE PROCEDURE find_replacement(
    p_id IN INT,
    p_date IN DATE
) IS
    profession_id INT;
    replacement_count INT := 0;

    -- User-defined exception
    no_replacement_found EXCEPTION;

    CURSOR c_replacements IS
        SELECT p.professional_id
        FROM Professional p
        WHERE p.prof_id = profession_id --the same profession
        AND p.professional_id != p_id --not him
        AND NOT EXISTS ( --doesn't have event in p_date
            SELECT 1
            FROM CustomerOrder co
            JOIN include i ON co.order_id = i.order_id
            JOIN Event e ON co.event_id = e.event_id
            WHERE i.professional_id = p.professional_id
            AND e.event_date = p_date
        );

BEGIN
    -- Get the profession ID of the original professional
    SELECT prof_id INTO profession_id
    FROM Professional
    WHERE professional_id = p_id;

    -- Print potential replacements
    DBMS_OUTPUT.PUT_LINE('Potential replacements:');
    FOR replacement IN c_replacements LOOP
        DBMS_OUTPUT.PUT_LINE('Professional ID: ' || replacement.professional_id);
        replacement_count := replacement_count + 1;
    END LOOP;

    -- Raise the user-defined exception if no replacements found
    IF replacement_count = 0 THEN
        RAISE no_replacement_found;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Total replacements found: ' || replacement_count);

EXCEPTION
    WHEN no_replacement_found THEN
        DBMS_OUTPUT.PUT_LINE('No replacement found. It's impossible to find a substitute.');
```

```
END;
```

דוגמת הרצה:

```
1 begin
2   -- Call the procedure
3   find_replacement(p_id => :p_id,
4                   p_date => :p_date);
5 end;
```

<input type="checkbox"/>	Variable	Type	Value
<input checked="" type="checkbox"/>	p_id	Float	5
<input checked="" type="checkbox"/>	p_date	Date	25/09/2022
<input checked="" type="checkbox"/>	*		

```
Professional ID: 189
Professional ID: 110
Professional ID: 245
Professional ID: 373
Professional ID: 122
Professional ID: 18
Professional ID: 153
Professional ID: 72
Professional ID: 92
Professional ID: 237
Professional ID: 177
Professional ID: 46
Professional ID: 198
Professional ID: 106
Professional ID: 324
Professional ID: 395
Professional ID: 227
Professional ID: 356
Professional ID: 285
Professional ID: 50
Professional ID: 118
Professional ID: 111
Professional ID: 48
Professional ID: 64
Professional ID: 330
Professional ID: 19
Professional ID: 7
Professional ID: 113
Professional ID: 216
Professional ID: 127
Professional ID: 353
Professional ID: 264
Professional ID: 257
Professional ID: 213
Professional ID: 274
Professional ID: 147
Professional ID: 158
Professional ID: 74
Professional ID: 40
Professional ID: 35
Professional ID: 289
Professional ID: 320
Professional ID: 279
Total replacements found: 184
```

פרוצדורה 2:

תיאור: הפרוצדורה מקבלת מספר מזהה של הזמנה, כמות לתשלום ושיטת תשלום. הפרוצדורה מוסיפה תשלום חדש להזמנה מסוימת. התשלום הוא התשלום שנותר לשלם עבור הזמנה זו. ומדפיסה את כל התשלומים הקשורים להזמנה זו.

```
1 CREATE OR REPLACE PROCEDURE add_payment(  
2     p_order_id IN NUMBER,  
3     p_amount IN NUMBER,  
4     p_pay_method_id IN NUMBER  
5 ) IS  
6     v_payment_id NUMBER;  
7 BEGIN  
8     -- Generate new payment_id  
9     SELECT NVL(MAX(payment_id), 0) + 1 INTO v_payment_id FROM Payment;  
10  
11    -- Insert new payment  
12    INSERT INTO Payment (payment_id, payment_amount, payment_date, pay_method_id, order_id)  
13    VALUES (v_payment_id, p_amount, SYSDATE, p_pay_method_id, p_order_id);  
14  
15    -- Print all payments for this order  
16    FOR payment_rec IN (SELECT * FROM Payment WHERE order_id = p_order_id) LOOP  
17        DBMS_OUTPUT.PUT_LINE('Payment ID: ' || payment_rec.payment_id ||  
18                               ', Amount: ' || payment_rec.payment_amount ||  
19                               ', Date: ' || payment_rec.payment_date);  
20    END LOOP;  
21  
22    COMMIT;  
23 EXCEPTION  
24 WHEN OTHERS THEN  
25     DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);  
26     ROLLBACK;  
27 END add_payment;
```

דוגמת הרצה:

```
1 begin  
2     -- Call the procedure  
3     add_payment(p_order_id => :p_order_id,  
4                 p_amount => :p_amount,  
5                 p_pay_method_id => :p_pay_method_id);  
6 end;
```

<input type="checkbox"/>	Variable	Type	Value
<input checked="" type="checkbox"/>	p_order_id	Float	5
<input checked="" type="checkbox"/>	p_amount	Float	21000
<input checked="" type="checkbox"/>	p_pay_method_id	Float	1

```
Payment ID: 5, Amount: 16000, Date: 29-אוגוסט-2023  
Payment ID: 400, Amount: 21000, Date: 04-יולי-2024
```

הוכחה מבסיס הנתונים:

<pre>SELECT payment_id, payment_amount, payment_date, pay_method_id, order_id FROM Payment WHERE order_id = 5 ORDER BY payment_date DESC;</pre>						
	PAYMENT_ID	PAYMENT_AMOUNT	PAYMENT_DATE	PAY_METHOD_ID	ORDER_ID	
1	400	21000	04/07/2024 16:57:47	1	5	
2	5	16000	29/08/2023	1	5	

תוכניות ראשיות

תוכנית 1:

תיאור: התוכנית מקבלת מזהה של בעל מקצוע ותאריך ובודקת האם הוא יכול לקחת חופש בתאריך זה. במידה ויש לו אירוע בתאריך המתקבל, מדפיסה את בעלי המקצוע שיכולים להחליף אותו אם קיימים כאלו. התוכנית קוראת לפונקציה ופרוצדורה שהגדרנו.

```
1 create or replace procedure manage_professional_schedule( p_id IN INT,
2   p_date IN DATE) is
3   is_available BOOLEAN;
4 BEGIN
5   DBMS_OUTPUT.PUT_LINE('Checking availability for Professional ID: ' || p_id || ' on date: ' || TO_CHAR(p_date, 'YYYY-MM-DD'));
6   DBMS_OUTPUT.PUT_LINE('---');
7
8   -- Check availability
9   is_available := check_p_availability(p_id, p_date);
10
11   -- If not available, find replacement
12   IF NOT is_available THEN
13     DBMS_OUTPUT.PUT_LINE('---');
14     DBMS_OUTPUT.PUT_LINE('Searching for a replacement...');
15     find_replacement(p_id, p_date);
16   END IF;
17
18   DBMS_OUTPUT.PUT_LINE('---');
19   DBMS_OUTPUT.PUT_LINE('Process completed.');
```

דוגמת הרצה:

1	begin
2	-- Call the procedure
3	manage_professional_schedule(p_id => :p_id,
4	p_date => :p_date);
5	end;

<input type="checkbox"/>	Variable	Type	Value
<input checked="" type="checkbox"/>	p_id	Float	5
<input checked="" type="checkbox"/>	p_date	Date	25/09/2022
<input checked="" type="checkbox"/>	*		

Checking availability for Professional ID: 5 on date: 2022-09-25

Professional is not available on 2022-09-25. Event details:

Event ID: 337

Event Date: 2022-09-25

Place: Banquet Hall

Order ID: 337

Total events found: 1

Searching for a replacement...

Potential replacements:

Professional ID: 379

Professional ID: 86

Professional ID: 352

Professional ID: 251

Professional ID: 28

Professional ID: 172

Professional ID: 311

Professional ID: 76

Professional ID: 99

Professional ID: 335

Professional ID: 25

Professional ID: 90

Professional ID: 31

Professional ID: 234

Professional ID: 357

Professional ID: 143

Professional ID: 358

Professional ID: 345

Professional ID: 337

Professional ID: 277

Professional ID: 365

Professional ID: 32

Professional ID: 304

Professional ID: 231

Professional ID: 346

Professional ID: 270

Professional ID: 183

Professional ID: 16

Professional ID: 33

Professional ID: 371

Professional ID: 284

Professional ID: 180

Professional ID: 20

Professional ID: 245

Professional ID: 373

Professional ID: 122

Professional ID: 18

Professional ID: 153

Professional ID: 72

Professional ID: 92

Professional ID: 237

Professional ID: 177

Professional ID: 46

Professional ID: 198

Professional ID: 106

Professional ID: 324

Professional ID: 395

Professional ID: 227

Professional ID: 356

Professional ID: 285

Professional ID: 50

Professional ID: 118

Professional ID: 111

Professional ID: 48

Professional ID: 64

Professional ID: 330

Professional ID: 19

Professional ID: 7

Professional ID: 113

Professional ID: 216

Professional ID: 127

Professional ID: 353

Professional ID: 264

Professional ID: 257

Professional ID: 213

Professional ID: 274

Professional ID: 147

Professional ID: 158

Professional ID: 74

Professional ID: 40

Professional ID: 35

Professional ID: 289

Professional ID: 320

Professional ID: 279

Total replacements found: 184

Process completed.

תוכנית 2:

תיאור: התוכנית עוברת על כל ההזמנות. עבור כל הזמנה מחשבת את התשלום הנותר לשלם על ידי שמפעילה את הפונקציה `calculate_remaining_payment`. מדפיסה את מזהה ההזמנה ושם הלקוח. אם נותר תשלום, מדפיסה את הסכום הנוסף שצריך לשלם, יוצרת שיטת תשלום אקראית, ומפעילה את הפרוצדורה `add_payment` כדי להוסיף תשלום חדש להזמנה. ומדפיסה הודעה אם ההזמנה שולמה במלואה

התוכנית מטפלת בחריגות, מדפיסה הודעה במקרה של תשלום יתר (מופעלת חריגה מותאמת אישית) או שגיאה בלתי צפויה אחרת.

```
1 CREATE OR REPLACE PROCEDURE process_all_orders IS
2     v_remaining NUMBER;
3     v_random_pay_method NUMBER;
4
5     -- Cursor to iterate through all orders
6     CURSOR c_orders IS
7         SELECT co.order_id, c.fname || ' ' || c.lname AS customer_name
8         FROM CustomerOrder co
9         JOIN Client c ON co.client_id = c.client_id;
10
11 BEGIN
12     FOR order_rec IN c_orders LOOP
13         BEGIN
14             -- Calculate remaining payment
15             v_remaining := calculate_remaining_payment(order_rec.order_id);
16
17             -- Print order information
18             DBMS_OUTPUT.PUT_LINE('Order ID: ' || order_rec.order_id);
19             DBMS_OUTPUT.PUT_LINE('Customer: ' || order_rec.customer_name);
20
21             IF v_remaining > 0 THEN
22                 DBMS_OUTPUT.PUT_LINE('Additional payment needed: ' || v_remaining);
23
24                 -- Generate random payment method (assuming 3 payment methods exist)
25                 v_random_pay_method := TRUNC(DBMS_RANDOM.VALUE(1, 4));
26
27                 -- Add new payment
28                 add_payment(order_rec.order_id, v_remaining, v_random_pay_method);
29
30                 DBMS_OUTPUT.PUT_LINE('New payment added successfully.');
```

```

1 begin
2     -- Call the procedure
3     process all orders;

```

```

Order ID: 371
Customer: Arnold Hornsby
Additional payment needed: 24000
Payment ID: 371, Amount: 21000, Date: 23-נובמבר-2021
Payment ID: 401, Amount: 24000, Date: 04-יולי-2024
New payment added successfully.
-----
An error occurred: ORA-20001: Overpayment detected. Refund amount: 14000
Order ID: 88
Customer: Arnold Hornsby
-----
An error occurred: ORA-20001: Overpayment detected. Refund amount: 6000
Order ID: 303
Customer: Bruce Baranski
-----
An error occurred: ORA-20001: Overpayment detected. Refund amount: 9000
Order ID: 144
Customer: Bruce Baranski
-----
An error occurred: ORA-20001: Overpayment detected. Refund amount: 3000
Order ID: 80
Customer: Bruce Baranski
-----
An error occurred: ORA-20001: Overpayment detected. Refund amount: 11000
Order ID: 372
Customer: Bryan Snipes
-----
An error occurred: ORA-20001: Overpayment detected. Refund amount: 35000
Order ID: 382
Customer: Rip Palmer
-----
Order ID: 241
Customer: Luis Saucedo
Additional payment needed: 1000
Payment ID: 241, Amount: 11000, Date: 19-יולי-2022
Payment ID: 402, Amount: 1000, Date: 04-יולי-2024
New payment added successfully.
-----
Order ID: 121
Customer: Patricia Tolkan
Additional payment needed: 27000
Payment ID: 121, Amount: 20000, Date: 08-ינואר-2023
Payment ID: 403, Amount: 27000, Date: 04-יולי-2024
New payment added successfully.
-----

```

Reverse engineering

תחילה לקחנו את קובץ ה createTable של הקבוצה השנייה, והמרנו אותה לסכמה, על מנת לראות את הקשרים באופן ברור.

קובץ ה createTable

```
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,
```

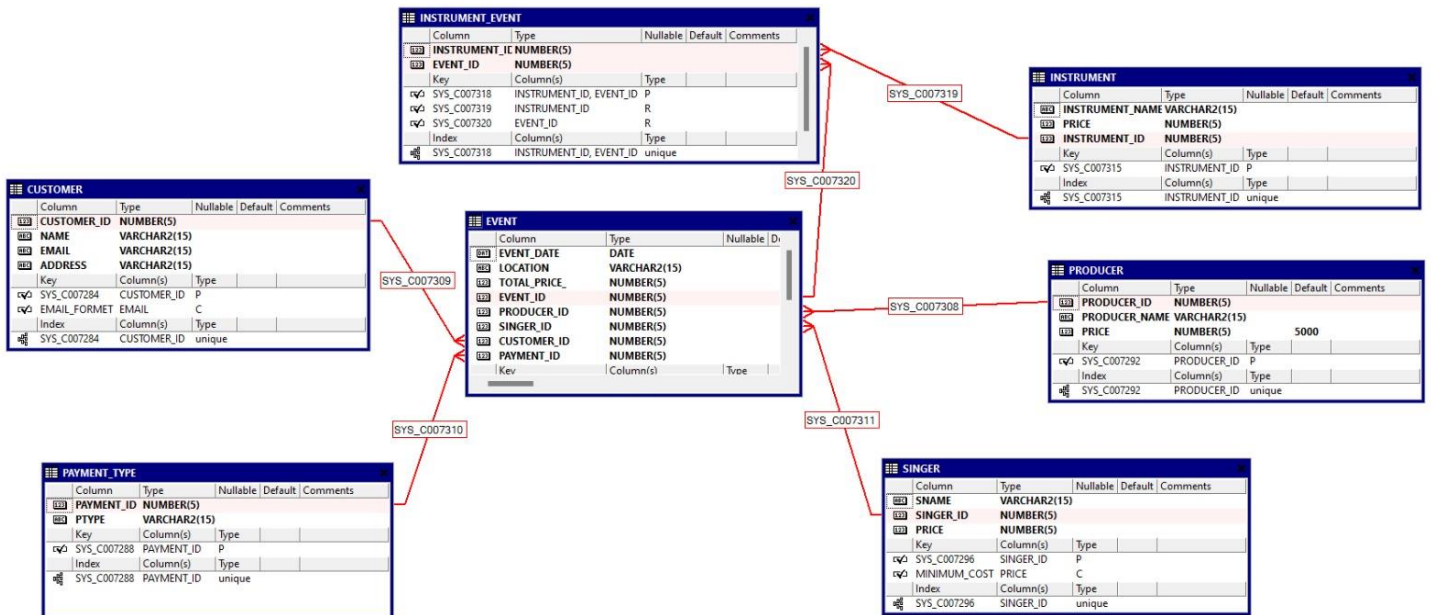
```

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 (Id) REFERENCES Singer(Id),
FOREIGN KEY (Customer_Id) REFERENCES Customer(Customer_Id),
FOREIGN KEY (Payment_Id) REFERENCES Payment_type(Payment_Id)
);

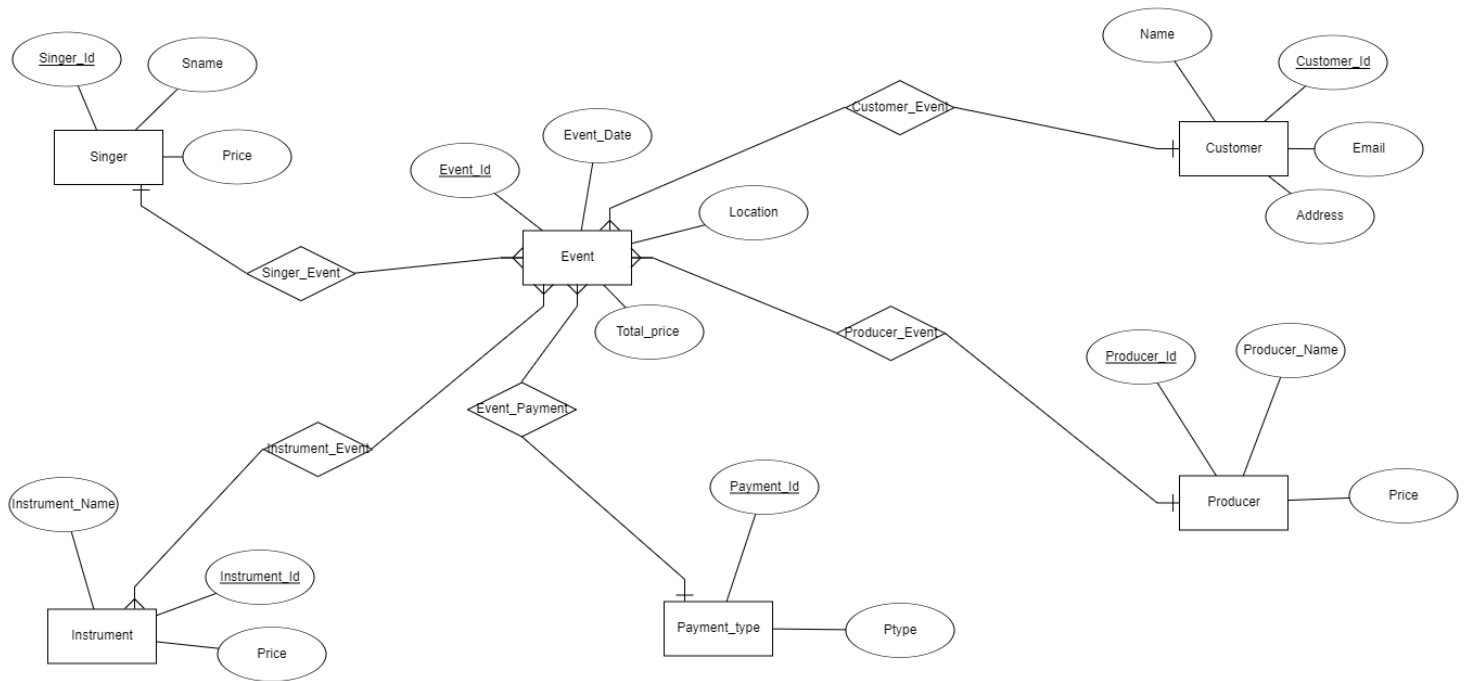
CREATE TABLE Instrument_Event
(
Instrument_Id NUMERIC(5) NOT NULL,
Event_Id NUMERIC(5) NOT NULL,
PRIMARY KEY (Instrument_Id, Event_Id),
FOREIGN KEY (Instrument_Id) REFERENCES Instrument(Instrument_Id),
FOREIGN KEY (Event_Id) REFERENCES Event(Event_Id)
);

```

יצרנו מהקובץ סכמת ה DSD - ע"י Diagram Window בתוכנה PL/SQL :

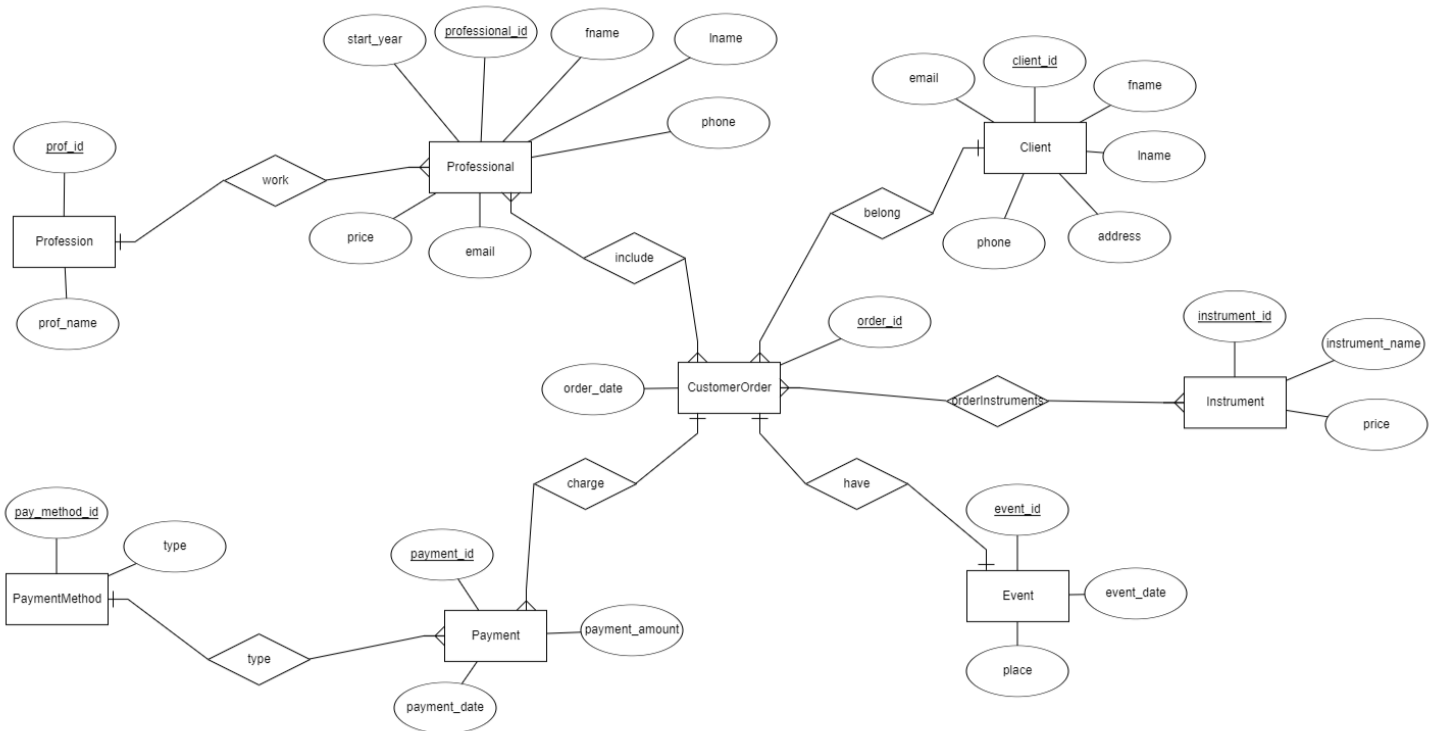


לסיים, ה ששחזרנו הוא:

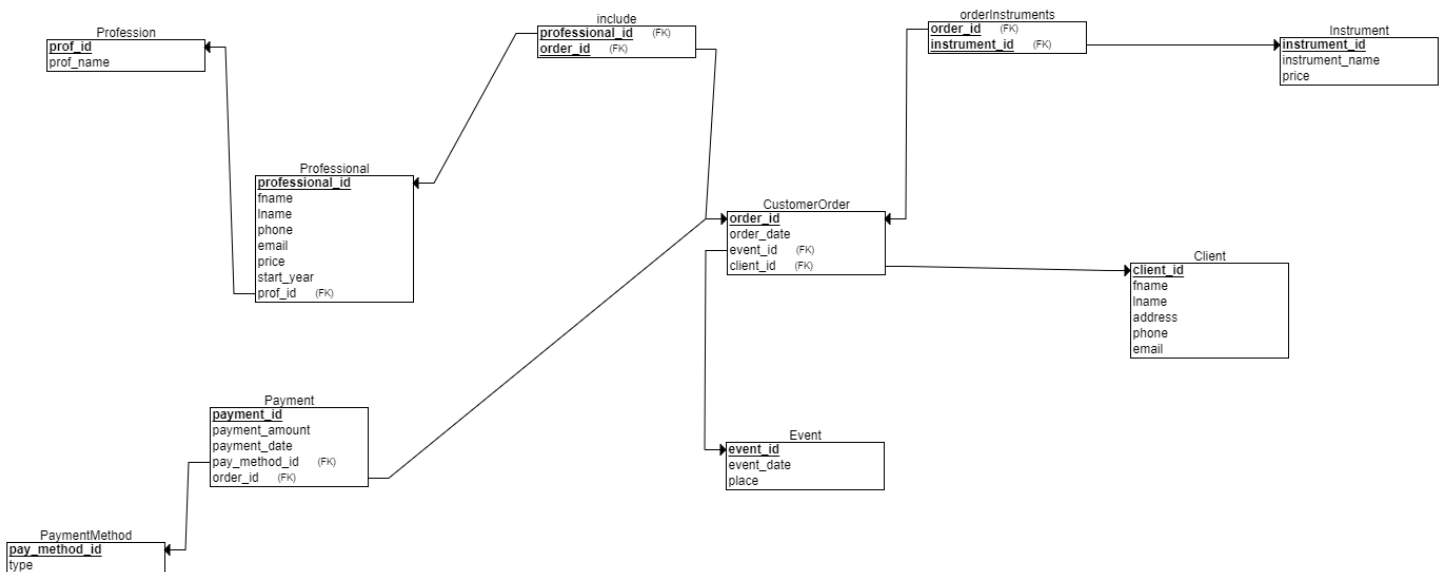


Integrated ERD

ה ERD המשולב:



ה DSD המשולב:



ההחלטות שקיבלנו:

1) הישויות המרכזיות המקושרות ל event ביחידה השנייה הינן singer producer בקשר של יחיד לרבים, כך שלהזמנה מקושר בדיוק זמר אחד ויוצר אחד. כדי להוסיף זמר ויוצר ליחידה שלנו, אין צורך בטבלאות של שניהם מהיחידה הראשונה, אלא יש להכניס לטבלת Profession את שני סוגי המקצועות הללו.

בכך חלים שני שינויים.

האחד, שבעלי המקצוע מקושרים להזמנה שמקושרת לאירוע (ולא ישירות להזמנה). השני, שניתן להזמין כמה בעלי מקצוע לאירוע (קשר של רבים לרבים).

2) כדי להוסיף תמיכה בכלי נגינה, הוספנו ישות Instrument כפי שמופיעה ביחידה השנייה בקשר של רבים לרבים לטבלת ה CustomerOrder (לא ישירות לאירוע)

3) אחדנו את הטבלאות Customer I Event מהיחידה השנייה עם הטבלאות Client I Event מהיחידה שלנו. לא היה צורך בהוספת שדות נוספים.

4) ביחידה שלנו הזמנה מקושרת בקשר של רבים לרבים לתשלום, ותשלום מקושר לסוג תשלום בקשר של רבים ליחיד. היחידה כפי שמתוארת מספקת תמיכה נרחבת יותר מאשר של היחידה השנייה, ולכן השארנו את הטבלאות הקשורות לתשלום כפי שהן ובכך אחדנו זאת עם טבלת ה Payment_type של יחידה זו.

אינטגרציה בבסיס הנתונים

- שינוי שמות הטבלאות של היחידה השנייה : הוספת סיומת '1' לשמות הטבלאות לצורך זיהוי ואבחנה.
 - הוספת שיטות תשלום חדשות : הוספת שיטות תשלום חדשות 'Bit' ו 'Check'-לטבלת PaymentMethod.
 - הוספת מקצועות חדשים : הוספת מקצועות 'Singer' ו 'Producer'-לטבלת Profession
 - הסרת אילוצי מפתח זר : הסרת אילוצי מפתח זר על עמודות Singer_Id ו Producer_Id -לטבלת Event1 כדי לאפשר עדכונים.
 - עדכון מזהים כדי למנוע התנגשויות : הגדלת הערכים של Singer_Id ב-500 ושל Producer_Id ב-1000 בטבלאות Singer1 ו Producer1-כדי למנוע התנגשויות עם מזהים קיימים.
 - העברת נתונים מטבלאות ישנות לחדשות : העברת נתונים מטבלאות Singer1 ו-Producer1 לטבלת Professional תוך שמירה על מקצועות חדשים, העברת נתוני אירועים מ Event1 - לטבלת Event והעברת נתוני לקוחות מ Customer1 - לטבלת Client רק אם המזהה לא קיים.
 - הוספת הזמנות ותשלומים : העברת נתוני אירועים מ Event1-לטבלת CustomerOrder , העברת נתוני תשלומים לטבלת Payment .
 - עדכון טבלת הכללה (Include) : הוספת קשרים בין הזמרים והיוצרים לאירועים בטבלת include.
 - עדכון מזהים ושמות טבלאות : הגדלת הערכים של Event_Id בטבלת Instrument_Event1 , שינוי שם הטבלה ל orderInstruments , שינוי שם העמודה ל event_id ל order_id - והוספת אילוץ מפתח זר לקישור עמודת order_id בטבלת orderInstruments לעמודת order_id בטבלת CustomerOrder .
- הסקריפט כולל גם שאילתות לבדיקת הנתונים אחרי העדכונים והעברות.
- להלן הסקריפט:

```
-- Renaming existing tables to add a suffix "1"
ALTER TABLE Singer RENAME TO Singer1;
ALTER TABLE Customer RENAME TO Customer1;
ALTER TABLE Instrument RENAME TO Instrument1;
ALTER TABLE Payment_type RENAME TO Payment_type1;
ALTER TABLE Producer RENAME TO Producer1;
ALTER TABLE Event RENAME TO Event1;
ALTER TABLE Instrument_Event RENAME TO Instrument_Event1;

-- Checking existing data in tables
SELECT * FROM PaymentMethod;
SELECT * FROM Payment_type1;

-- Inserting new payment methods
INSERT INTO PaymentMethod (pay_method_id, type)
VALUES (4, 'Bit'); -- Adding 'Bit' payment method
INSERT INTO PaymentMethod (pay_method_id, type)
VALUES (5, 'Check'); -- Adding 'Check' payment method
```

```

-- Checking existing data in tables
SELECT * FROM Profession;

-- Inserting new professions
INSERT INTO Profession (prof_id, prof_name)
VALUES (2, 'Singer'); -- Adding 'Singer' profession
INSERT INTO Profession (prof_id, prof_name)
VALUES (3, 'Producer'); -- Adding 'Producer' profession

-- Checking existing data in tables
SELECT * FROM Professional;
SELECT * FROM Customer1;
SELECT * FROM Client;

-- Dropping foreign key constraints on Event1 table
ALTER TABLE Event1
DROP CONSTRAINT SYS_C007308; -- Dropping FK constraint on Singer_Id
ALTER TABLE Event1
DROP CONSTRAINT SYS_C007311; -- Dropping FK constraint on Producer_Id

-- Updating IDs in Singer1 and Producer1 tables to avoid conflicts
UPDATE Singer1
SET Singer_Id = Singer_Id + 500; -- Updating Singer_Id
UPDATE Producer1
SET Producer_Id = Producer_Id + 1000; -- Updating Producer_Id

-- Checking updated data in tables
SELECT * FROM Singer1;

-- Transferring data from Singer1 and Producer1 to Professional
INSERT INTO Professional (professional_id, fname, price, start_year,
prof_id)
SELECT Singer_Id, SNAME, Price, 2000, 2
FROM Singer1; -- Inserting singers into Professional
INSERT INTO Professional (professional_id, fname, price, start_year,
prof_id)
SELECT Producer_Id, Producer_NAME, Price, 2000, 3
FROM Producer1; -- Inserting producers into Professional

-- Checking updated data in tables
SELECT * FROM Producer1;

-- Updating IDs in Event1 table to avoid conflicts
UPDATE Event1
SET Singer_Id = Singer_Id + 500,
    Producer_Id = Producer_Id + 1000; -- Updating Singer_Id and
Producer_Id in Event1

-- Dropping foreign key constraint on Instrument_Event1 table
ALTER TABLE Instrument_Event1
DROP CONSTRAINT SYS_C007320; -- Dropping FK constraint on Event_Id

-- Updating Event_Id in Event1 table to avoid conflicts
UPDATE Event1
SET Event_Id = Event_Id + 500; -- Updating Event_Id in Event1

-- Transferring data from Event1 to Event
INSERT INTO Event (event_id, event_date, place)
SELECT Event_Id, Event_Date, Location
FROM Event1; -- Inserting events into Event

```

```

-- Checking updated data in tables
SELECT * FROM Event;

-- Transferring unique customers from Customer1 to Client
INSERT INTO Client (client_id, fname, address, email)
SELECT c.Customer_Id, c.name, c.address, c.email
FROM Customer1 c
WHERE NOT EXISTS (
    SELECT 1
    FROM Client cl
    WHERE cl.client_id = c.Customer_Id
); -- Inserting unique customers into Client

-- Transferring data from Event1 to CustomerOrder
INSERT INTO CustomerOrder (order_id, order_date, event_id, client_id)
SELECT Event_Id, Event_Date, Event_Id, Customer_Id
FROM Event1; -- Inserting orders into CustomerOrder

-- Transferring payment data from Event1 to Payment
INSERT INTO Payment (payment_id, payment_amount, payment_date,
pay_method_id, order_id)
SELECT Event_Id, Total_price_, Event_Date, Payment_Id-1, Event_Id
FROM Event1; -- Inserting payments into Payment

-- Checking updated data in tables
SELECT * FROM Payment;

-- Inserting singers and producers into include table
INSERT INTO include (professional_id, order_id)
SELECT Singer_Id, Event_Id
FROM Event1; -- Inserting singers into include
INSERT INTO include (professional_id, order_id)
SELECT Producer_Id, Event_Id
FROM Event1; -- Inserting producers into include

-- Checking updated data in tables
SELECT * FROM Include;

-- Updating Event_Id in Instrument_Event1 and renaming to
orderInstruments
UPDATE Instrument_Event1
SET Event_Id = Event_Id + 500; -- Updating Event_Id in
Instrument_Event1

-- Renaming table Instrument_Event1 to orderInstruments
ALTER TABLE Instrument_Event1
RENAME TO orderInstruments; -- Renaming Instrument_Event1 to
orderInstruments

-- Renaming column event_id to order_id in orderInstruments
ALTER TABLE orderInstruments
RENAME COLUMN event_id TO order_id; -- Renaming event_id to order_id
in orderInstruments

-- Adding foreign key constraint to orderInstruments
ALTER TABLE orderInstruments
ADD CONSTRAINT fk_instrument_order
FOREIGN KEY (order_id)
REFERENCES CustomerOrder (order_id); -- Adding FK constraint on
order_id in orderInstruments

```

View

View 1: עבור היחידה שלנו.

במבט זה יצרנו טבלה של קישור הזמנה עם בעלי המקצוע שהוזמנו לה.

SQL	Output	Statistics
<pre>-- Creating a view named OrderProfessionalView -- This view displays order details along with professional details and their professions CREATE VIEW OrderProfessionalView AS SELECT co.order_id, -- Order ID co.order_date, -- Order Date p.fname AS professional_first_name, -- Professional's First Name p.lname AS professional_last_name, -- Professional's Last Name p.price AS professional_price, -- Professional's Price pr.prof_name AS profession_name -- Profession Name of the Professional FROM CustomerOrder co JOIN include i ON co.order_id = i.order_id JOIN Professional p ON i.professional_id = p.professional_id JOIN Profession pr ON p.prof_id = pr.prof_id;</pre>		

שאלתה 1:

מחשבת את העלות של כל הזמנה לפי מחיר בעלי המקצוע שהוזמנו עבורה, תוך כדי התעלמות מבעלי המקצוע זמר ויוצר שנוספו מהיחידה השנייה.

```
-- Query to calculate the total price of each order, excluding orders that include professions with IDs 2 and 3
-- Returns the total price for each order
SELECT
    order_id,
    SUM(professional_price) AS total_order_price
FROM
    OrderProfessionalView
WHERE
    order_id NOT IN (
        SELECT DISTINCT order_id
        FROM OrderProfessionalView
        WHERE profession_id IN (2, 3) -- Excluding orders with professions SINGER and PRODUCER
    )
GROUP BY
    order_id; -- Grouping by order ID
```

שאלתה 2:

מחזירה את מספר ההזמנות הקיימות עבור כל סוג מקצוע. המקצוע המשתתף במספר ההזמנות הרב ביותר יופיע ראשון.

```
-- Query to return the profession name and the number of orders for each profession, excluding professions with IDs 2 and 3
-- The profession with the highest number of orders is listed first
SELECT
    profession_name,|
    profession_id,
    COUNT(order_id) AS number_of_orders
FROM
    OrderProfessionalView
WHERE
    profession_id NOT IN (2, 3) -- Excluding professions SINGER and PRODUCER
GROUP BY
    profession_name,
    profession_id
ORDER BY
    number_of_orders DESC; -- Ordering by number of orders
```

View 2: עבור היחידה השנייה.

במבט זה יצרנו טבלה של קישור הזמנה עם האירוע התואם לה, הלקוח שלה, וכלי הנגינה שהוזמנו לה.

```
-- Creating a view that consolidates data from multiple tables related to orders and instruments
CREATE VIEW OrderInstrumentView AS
SELECT
    e.event_id,
    e.event_date,
    e.place,
    c.client_id,
    c.fname AS client_first_name,
    c.lname AS client_last_name,
    co.order_id,
    co.order_date,
    i.instrument_id,
    i.instrument_name,
    i.price AS instrument_price
FROM
    Event e
JOIN
    CustomerOrder co ON e.event_id = co.event_id
JOIN
    Client c ON co.client_id = c.client_id
JOIN
    orderInstruments oi ON co.order_id = oi.order_id
JOIN
    Instrument i ON oi.instrument_id = i.instrument_id;
```

שאלתה 1:

מציגה עבור כל לקוח את מספר ההזמנות שלו והמחיר הכולל של כלי הנגינה בהזמנותיו.

```
-- Query to get the total number of distinct orders and total instrument price for each client,  
-- ordered by the number of orders in descending order.  
SELECT  
    client_id,  
    client_first_name,  
    client_last_name,  
    COUNT(DISTINCT order_id) AS number_of_orders,  
    SUM(instrument_price) AS total_instrument_price  
FROM  
    OrderInstrumentView  
GROUP BY  
    client_id,  
    client_first_name,  
    client_last_name  
ORDER BY  
    number_of_orders DESC;
```

שאלתה 2:

מציגה את סך הפעמים שהוזמן כל כלי נגינה. (בסדר יורד)

```
-- Query to get the number of occurrences of each instrument in orders,  
-- ordered by the number of occurrences in descending order.  
SELECT  
    instrument_id,  
    instrument_name,  
    COUNT(order_id) AS number_of_occurrences  
FROM  
    OrderInstrumentView  
GROUP BY  
    instrument_id,  
    instrument_name  
ORDER BY  
    number_of_occurrences DESC;
```