

Step B

1. Introduction:

This report documents the work completed for Step B of the Database Project. The project involves the design, implementation, and manipulation of a relational database system. This stage focuses on querying the database, modifying its content, and enforcing constraints to ensure data integrity.

2. Objectives:

The primary objectives of Step B are:

- **Querying the Database:** To retrieve specific information from the database using complex SQL queries. These queries provide insights into sales trends, customer behaviour, employee performance, and product inventory.
- **Data Modification:** To update and delete specific records in the database to maintain accurate and up-to-date information.
- **Enforcing Constraints:** To ensure data integrity by adding constraints to the database schema and demonstrating the effectiveness of these constraints through practical examples.

3. Scope:

The scope of this step includes:

- **Select Queries:**

Four select queries without parameters that provide detailed insights into various aspects of the database.

Four select queries with parameters, demonstrating the use of dynamic filtering and querying techniques.

- **Update and Delete Queries:**

Two update queries that modify existing data in the database.

Two delete queries that remove specific records from the database.

- **Constraints:**

Adding and enforcing constraints to ensure data accuracy and integrity.

4. Queries.sql

Query 1: Select the product with the highest price for each type of product. List the products in descending order of price.

שאלתה זו בוחרת את המוצר עם המחיר הגבוה ביותר עבור כל סוג של מוצר ומפרטת אותם בסדר יורד של המחיר. זה עוזר לזהות את המוצר היקר ביותר בכל קטגוריה.

```
SELECT p.ProductID, p.ProductName, p.Price, p.TypeOfProduct
FROM Products p
WHERE p.Price = (
    SELECT MAX(p2.Price)
    FROM Products p2
    WHERE p2.TypeOfProduct = p.TypeOfProduct
)
ORDER BY p.Price DESC;
```

Screenshot of the result

PRODUCTID	PRODUCTNAME	PRICE	TYPEOFPRODUCT
10398	Skin Care	543	Electronics
10311	Laptop	494	Sports Equipment
10386	Skin Care	492	Beauty
10221	Laptop	492	Pet Supplies

Query 2: Group the sales by year and month, showing the total sales amount and average sales amount for each month.

שאלתה זו מקבצת את המכירות לפי שנה וחודש, ומציגה את כמות המכירות הכוללת וסכום המכירות הממוצע עבור כל חודש. היא מספקת תובנות לגבי ביצועי מכירות חודשיים.

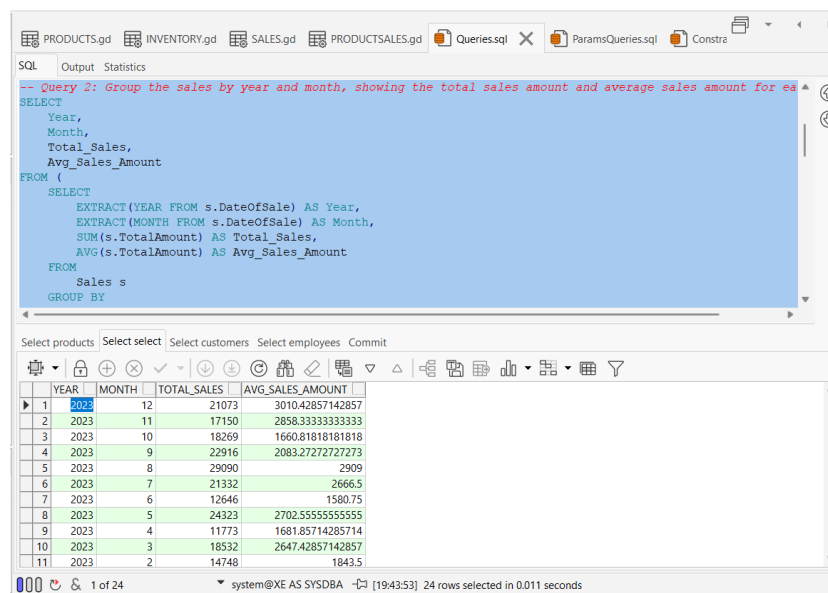
```

SELECT
    Year,
    Month,
    Total_Sales,
    Avg_Sales_Amount
FROM (
    SELECT
        EXTRACT(YEAR FROM s.DateOfSale) AS Year,
        EXTRACT(MONTH FROM s.DateOfSale) AS Month,
        SUM(s.TotalAmount) AS Total_Sales,
        AVG(s.TotalAmount) AS Avg_Sales_Amount
    FROM
        Sales s
    GROUP BY
        EXTRACT(YEAR FROM s.DateOfSale),
        EXTRACT(MONTH FROM s.DateOfSale)
) MonthlyStats
ORDER BY
    Year DESC,

    Month DESC;

```

Screenshot of the result



	YEAR	MONTH	TOTAL_SALES	AVG_SALES_AMOUNT
1	2023	12	21073	3010.42857142857
2	2023	11	17150	2858.33333333333
3	2023	10	18269	1660.81818181818
4	2023	9	22916	2083.27272727273
5	2023	8	29090	2909
6	2023	7	21332	2666.5
7	2023	6	12646	1580.75
8	2023	5	24323	2702.55555555555
9	2023	4	11773	1681.85714285714
10	2023	3	18532	2647.42857142857
11	2023	2	14748	1843.5

Query 3: List customers who have made purchases totalling more than \$5,000 in the last year. Order by total amount spent.

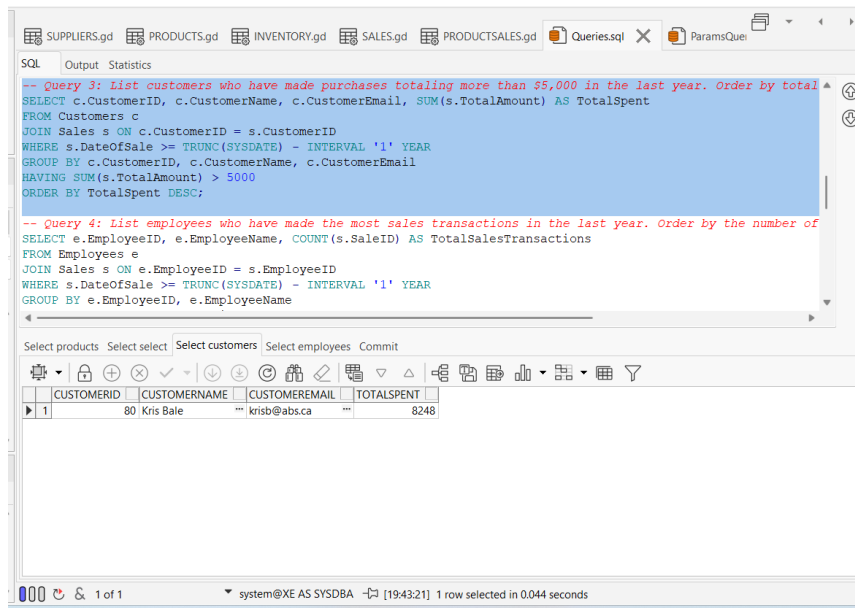
שאלתה זו מפרטת לקוחות שביצעו רכישות בסכום כולל של יותר מ-\$5,000 בשנה האחרונה, לפי הסכום הכולל שהוצא. זה עוזר לזהות לקוחות בעלי הוצאה גבוהה.

```

SELECT c.CustomerID, c.CustomerName, c.CustomerEmail, SUM(s.TotalAmount) AS
TotalSpent
FROM Customers c
JOIN Sales s ON c.CustomerID = s.CustomerID
WHERE s.DateOfSale >= TRUNC(SYSDATE) - INTERVAL '1' YEAR
GROUP BY c.CustomerID, c.CustomerName, c.CustomerEmail
HAVING SUM(s.TotalAmount) > 5000
ORDER BY TotalSpent DESC;

```

Screenshot of the result



Query 4: List employees who have made the most sales transactions in the last year. Order by the number of sales transactions in descending order.

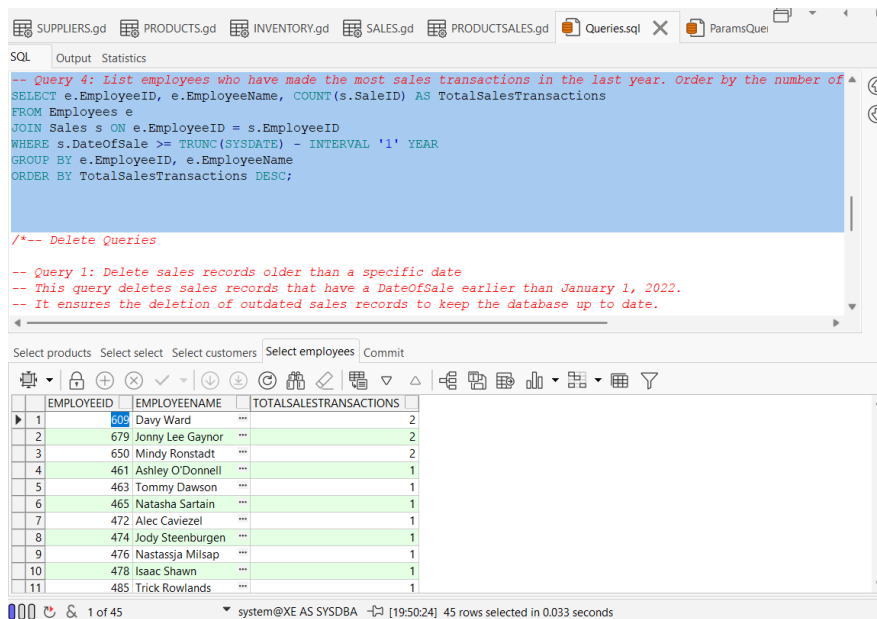
שאלתה זו מפרטת עובדים שביצעו את מירב עסקאות המכירה בשנה האחרונה, מסודרים לפי מספר עסקאות המכירה בסדר יורד. זה מדגיש את העובדים עם הביצועים הטובים ביותר במונחים של נפח מכירות.

```

SELECT e.EmployeeID, e.EmployeeName, COUNT(s.SaleID) AS
TotalSalesTransactions
FROM Employees e
JOIN Sales s ON e.EmployeeID = s.EmployeeID
WHERE s.DateOfSale >= TRUNC(SYSDATE) - INTERVAL '1' YEAR
GROUP BY e.EmployeeID, e.EmployeeName
ORDER BY TotalSalesTransactions DESC;

```

Screenshot of the result

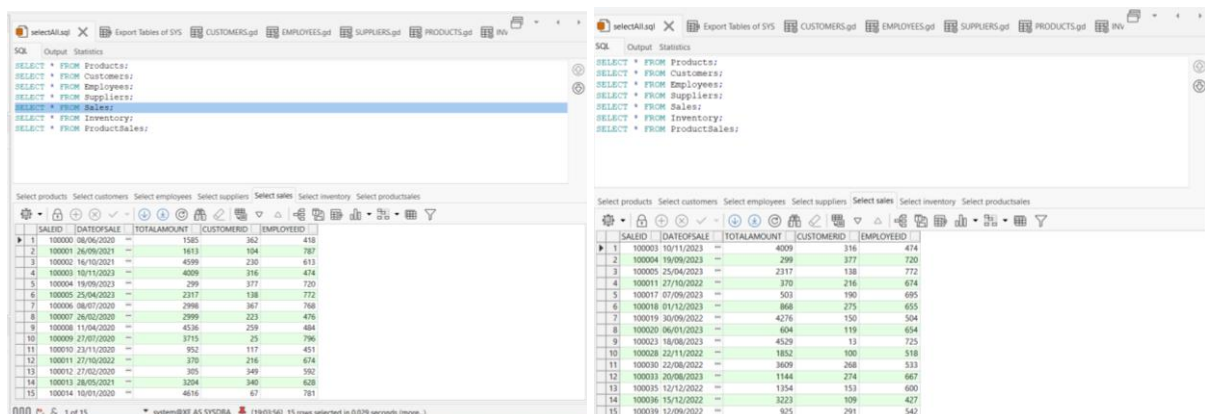


Delete Query 1: Delete sales records older than a specific date.

שאלתה זו מוחקת רשומות מכירות שיש להן DateOfSale מוקדם יותר מ-1 בינואר 2022. היא מבטיחה מחיקה של רשומות מכירות מיושנות כדי לשמור על עדכניות מסד הנתונים.

```
DELETE FROM Sales
WHERE DateOfSale < TO_DATE('01-01-2022', 'DD-MM-YYYY');
```

Screenshot of the database before and after the update



Delete Query 2: Delete inventory records where quantity is zero.

שאלתה זו מוחקת רשומות מלאי שבהן הכמות היא אפס. זה מנקה את טבלת המלאי על ידי הסרת רשומות שמצביעות על אין מלאי.

```
DELETE FROM Inventory
WHERE Quantity = 0;
```

Screenshot of the database before and after the update:

The screenshots show the SQL Server Enterprise Manager interface. The left screenshot shows the 'Inventory' table with 192 rows. The right screenshot shows the 'Inventory' table after the update, with 192 rows remaining. The status bar at the bottom of the right screenshot indicates '192 rows selected in 0.045 seconds (more...)'. The status bar at the bottom of the left screenshot indicates '187 of 400'.

Update Query 1: Update the position of employees who are currently 'Cashier' to 'Janitor'.

שאלתה זו מעדכנת את עמדת העובדים שהם כרגע "Cashier" ל-"Janitor".

```
UPDATE Employees
SET Position = 'Janitor'
WHERE Position = 'Cashier';
```

Screenshot of the database before and after the update

The screenshots show the SQL Server Enterprise Manager interface. The left screenshot shows the 'Employees' table with 15 rows. The right screenshot shows the 'Employees' table after the update, with 15 rows remaining. The status bar at the bottom of the right screenshot indicates '15 rows selected in 0.045 seconds (more...)'. The status bar at the bottom of the left screenshot indicates '3 of 15'.

Update Query 2: Update the price of all products by increasing it by 10% for products in the 'Electronics' category.

שאלתה זו מעדכנת את המחיר של כל המוצרים בקטגוריית "Electronics" על ידי העלאתו ב-10%.

```
UPDATE Products
SET Price = Price * 1.10
WHERE TypeOfProduct = 'Electronics';
```

Screenshot of the database before and after the update

PRODUCTID	PRODUCTNAME	PRICE	TYPEOFPRODUCT
1	10000 Dog Food	314	Electronics
2	10001 Skin Care	262	Electronics
3	10002 Skin Care	397	Pet Supplies
4	10003 Sports Shoes	273	Pet Supplies
5	10004 Phone	365	Pet Supplies
6	10005 Dog Food	418	Sports Equipment
7	10006 Phone	400	Electronics
8	10007 Skin Care	126	Beauty
9	10008 Phone	489	Sports Equipment
10	10009 Phone	332	Beauty
11	10010 Skin Care	182	Pet Supplies
12	10011 Skin Care	212	Beauty
13	10012 Laptop	414	Electronics
14	10013 Phone	330	Beauty
15	10014 Laptop	488	Beauty

5. ParamsQueries.sql

```
-- Query 4: Select inventory items restocked by a specified supplier
-- Parameters: SupplierName, RestockDate
SELECT p.ProductName, s.SupplierName, i.Quantity, i.LastRestockedDate
FROM Inventory i
JOIN Products p ON i.ProductID = p.ProductID
JOIN Suppliers s ON i.SupplierID = s.SupplierID
WHERE s.SupplierName = '&SupplierName'
AND i.LastRestockedDate > TO_DATE('&RestockDate', 'DD-MM-YYYY')
ORDER BY i.LastRestockedDate DESC;
```

Query 1: Select customers who made purchases after a specified date.

שאלתה זו בוחרת לקוחות שביצעו רכישות לאחר תאריך מוגדר, לפי הסכום הכולל שהוצא. הפרמטר StartDate משמש לסינון התוצאות.

```
-- Parameter: StartDate (DATE)
SELECT c.CustomerName, c.CustomerEmail, SUM(s.TotalAmount) AS TotalSpent
FROM Customers c
JOIN Sales s ON c.CustomerID = s.CustomerID
WHERE s.DateOfSale > TO_DATE('&StartDate', 'DD-MM-YYYY')
GROUP BY c.CustomerID, c.CustomerName, c.CustomerEmail

ORDER BY TotalSpent DESC;
```

Screenshot of the result

The screenshot shows a database query tool with two queries. Query 1 is highlighted in blue and shows the SQL for selecting customers by total spent. Query 2 is for selecting employees by total revenue. Below the queries is a table of results for Query 1.

	CUSTOMERNAME	CUSTOMEREMAIL	TOTALSPENT
1	Kris Bale	krisb@abs.ca	8248
2	Jena Allan	jena@sfgo.jp	6045
3	Stevie Pleasence	stevie.pleasence@directdata.il	5625
4	Kazem Larter	kazem.larter@cynergydata.uk	4939
5	Bryan Cube	bryan.cube@telwares.il	4786
6	Thomas Henstridge	thomas.henstridge@infinity.ie	4542
7	Gil Preston	g.preston@myricom.de	4511
8	Dianne Englund	denglund@timevision.de	4439
9	Dom Adler	dom.adler@northhighland.no	4371
10	Jonatha McGregor	jonatha.mcgregor@kelmooreinvestment.com	4259
11	Nickv Schiff	nickv.s@marsinc.cn	4243

Query 2: Select employees who made sales of a specified minimum amount.

שאלתה זו בוחרת עובדים שביצעו מכירות עם סכום כולל מינימלי מוגדר, מסודר לפי סך ההכנסות. הפרמטר MinAmount משמש לסינון התוצאות.

```
-- Parameter: MinAmount (NUMBER)
SELECT e.EmployeeName, COUNT(s.SaleID) AS TotalSales, SUM(s.TotalAmount) AS TotalRevenue
FROM Employees e
JOIN Sales s ON e.EmployeeID = s.EmployeeID
GROUP BY e.EmployeeID, e.EmployeeName
HAVING SUM(s.TotalAmount) > &<name=MinAmount default=5000 required=true>

ORDER BY TotalRevenue DESC;
```


Screenshot of the result

SQL Output Statistics

```
-- Query 2: Select employees who made sales of a specified minimum amount.
-- Parameter: MinAmount (NUMBER)
SELECT e.EmployeeName, COUNT(s.SaleID) AS TotalSales, SUM(s.TotalAmount) AS TotalRevenue
FROM Employees e
JOIN Sales s ON e.EmployeeID = s.EmployeeID
GROUP BY e.EmployeeID, e.EmployeeName
HAVING SUM(s.TotalAmount) > <name=MinAmount default=5000 required=true>
ORDER BY TotalRevenue DESC;
```

```
-- Query 3: Select products within a specified price range and a specified type.
-- Parameters: MinPrice, MaxPrice, ProductType
SELECT p.ProductName, p.Price, p.TypeOfProduct
FROM Products p
WHERE p.Price BETWEEN <name=MinPrice default=400 required=true>
AND <name=MaxPrice default=500 required=true>
AND p.TypeOfProduct = '<name=ProductType list="select distinct TypeOfProduct from Products" required=true>'
```

Select customers Select employees Select products Select inventory Commit

	EMPLOYEENAME	TOTALSALES	TOTALREVENUE
1	Jonny Lee Gaynor	3	10593
2	Ian Thomson	3	9045
3	Gene Hanley	2	8757
4	Kris Kimball	3	8597
5	Robbie Boyle	2	6692
6	Rhys Duschel	2	6467
7	Alex Bratt	2	6330
8	Ramsey Cruz	2	6005
9	Paula Sharp	2	5955
10	Lucinda Hudson	2	5854
11	Dawn Ward	3	5775

system@XE AS SYSDBA [19:53:02] 14 rows selected in 0.040 seconds

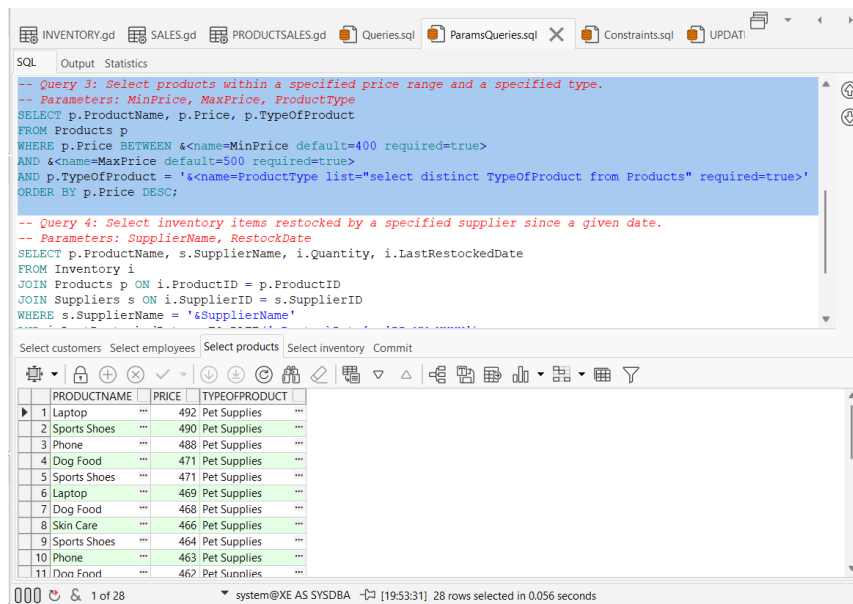
Query 3: Select products within a specified price range and a specified type.

שאלתה זו בוחרת מוצרים בטווח מחירים מוגדר ומסוג מוגדר, לפי מחיר. הפרמטרים MinPrice, MaxPrice ו- ProductType משמשים לסינון התוצאות.

```
-- Parameters: MinPrice, MaxPrice, ProductType
SELECT p.ProductName, p.Price, p.TypeOfProduct
FROM Products p
WHERE p.Price BETWEEN <name=MinPrice default=400 required=true>
AND <name=MaxPrice default=500 required=true>
AND p.TypeOfProduct = '<name=ProductType list="select distinct
TypeOfProduct from Products" required=true>'

ORDER BY p.Price DESC;
```

Screenshot of the result

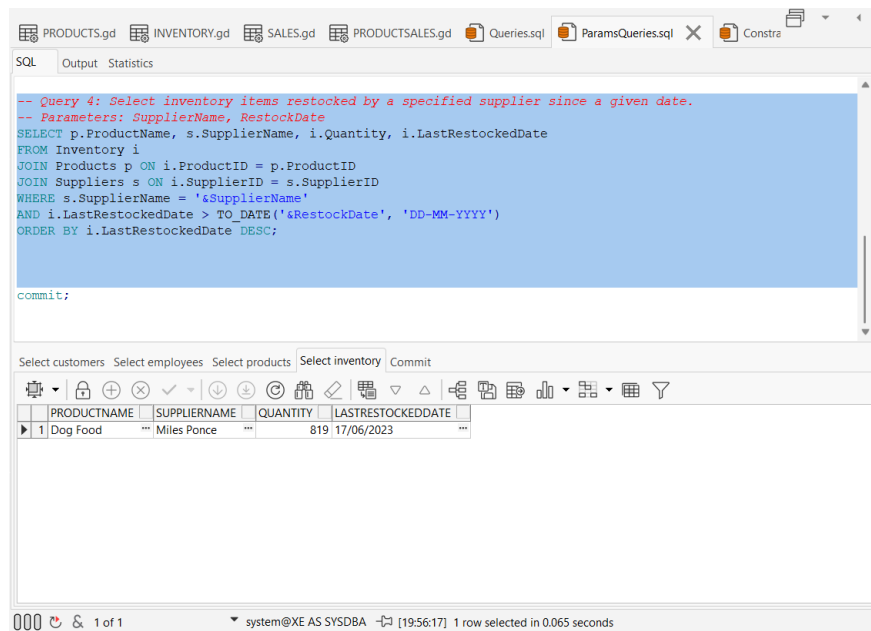


Query 4: Select inventory items restocked by a specified supplier since a given date.

שאלתה זו בוחרת פריטי מלאי שהוחזרו על ידי ספק שצוין מאז תאריך נתון, מסודרים לפי תאריך החידוש האחרון של המלאי. הפרמטרים SupplierName ו-RestockDate משמשים לסינון התוצאות.

```
-- Parameters: SupplierName, RestockDate
SELECT p.ProductName, s.SupplierName, i.Quantity, i.LastRestockedDate
FROM Inventory i
JOIN Products p ON i.ProductID = p.ProductID
JOIN Suppliers s ON i.SupplierID = s.SupplierID
WHERE s.SupplierName = '&SupplierName'
AND i.LastRestockedDate > TO_DATE('&RestockDate', 'DD-MM-YYYY')
ORDER BY i.LastRestockedDate DESC;
```

Screenshot of the result



6. Constraints.sql

Constraint 1: Add a UNIQUE constraint to the CustomerEmail column in the Customers table.

פקודה זו מוסיפה אילוץ ייחודי לעמודה CustomerEmail בטבלת הלקוחות, ומבטיחה שלשני לקוחות לא תהיה אותה כתובת דוא"ל.

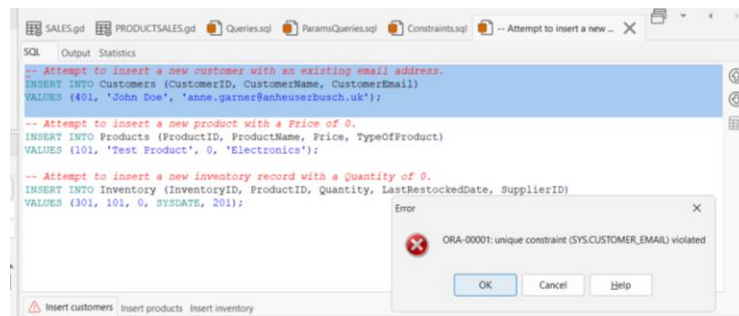
```
ALTER TABLE Customers
```

```
ADD CONSTRAINT customer_email UNIQUE (CustomerEmail);
```

Attempt to insert data that violates the constraint

```
INSERT INTO Customers (CustomerID, CustomerName, CustomerEmail)
VALUES (401, 'John Doe', 'anne.garner@anheuserbusch.uk');
```

Screenshot showing the run-time error



Constraint 2: Add a CHECK constraint to ensure the Price in the Products table is greater than 0.

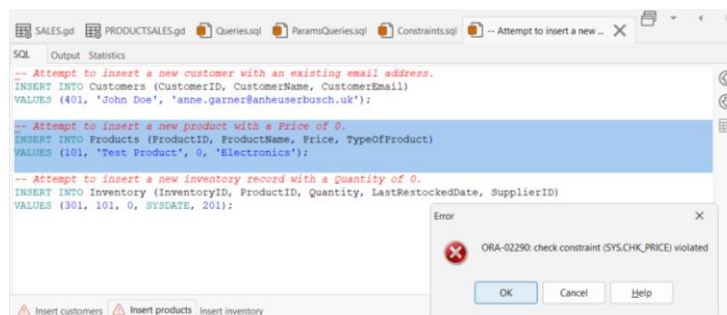
פקודה זו מוסיפה אילוץ CHECK כדי להבטיח שהמחיר בטבלת המוצרים גדול מ-0.

```
ALTER TABLE Products
ADD CONSTRAINT chk_price CHECK (Price > 0);
```

Attempt to insert data that violates the constraint

```
INSERT INTO Products (ProductID, ProductName, Price, TypeOfProduct)
VALUES (101, 'Test Product', 0, 'Electronics');
```

Screenshot showing the run-time error



Constraint 3: Add a CHECK constraint to ensure the Quantity in the Inventory table is greater than 0.

פקודה זו מוסיפה אילוץ CHECK כדי להבטיח שהכמות בטבלת המלאי גדולה מ-0.

```
ALTER TABLE Inventory
```

```
ADD CONSTRAINT chk_quantity CHECK (Quantity > 0);
```

Attempt to insert data that violates the constraint

```
INSERT INTO Inventory (InventoryID, ProductID, Quantity, LastRestockedDate,  
SupplierID)  
VALUES (301, 101, 0, SYSDATE, 201);
```

Screenshot showing the run-time error

