

תשובות

חלק א

1. כתבו שאילתה המציגה את הכמות, סכום המכירות וכמות החשבוניות לכל פריט.

SQLQuery5.s....77) (משתמש)* SQLQuery4....55) (משתמש)* X SQLQuery3.s....71) (משתמש)*

```

1 SELECT
2     ItemCode,
3     SUM(Qty) AS TotalQuantitySold,
4     SUM(LineSum) AS TotalSalesAmount,
5     COUNT(DISTINCT DocNum) AS InvoiceCount
6 FROM SalesLine
7 GROUP BY ItemCode;
8

```

150 % 6 0 ↑ ↓

Results Messages

	ItemCode	TotalQuantitySold	TotalSalesAmount	InvoiceCount
1	1001	3	150.00	2
2	1002	2	110.00	2
3	1003	4	220.00	2
4	3611010	4	260.00	3
5	3611600	4	300.00	3

2. כתבו שאילתה המציגה את מספרי החשבוניות שבכל אחת מהן קיים פריט 3611010 וגם פריט 3611600.

```

1 SELECT DocNum
2 FROM SalesLine
3 WHERE ItemCode IN (3611010, 3611600)
4 GROUP BY DocNum
5 HAVING COUNT(DISTINCT ItemCode) = 2;
6
7

```

150 % 5 0 ↑ ↓

Results Messages

	DocNum
1	103
2	105

3. כתבו שאילתה המציגה את אנשי המכירות שמכרו את כלל הפריטים המוצעים בקטלוג.

SQLQuery5.....77) * (ששתחש) * X | (ששתחש) SQLQuery4.s...55) * (ששתחש) * | (ששתחש) SQLQuery3.s.../1) * (ששתחש) * | DESKTOP-7898... - dbo.Items | (ששתחש) SQLQuery2.s...59) * (ששתחש) *

```
1 SELECT SalesPersonName FROM SalesPerson AS SP
2 JOIN SalesHeader AS SH
3 ON SP.SalesPersonCode=SH.SalesPersonCode
4 JOIN SalesLine AS SL
5 ON SH.DocNum=SL.DocNum
6 JOIN Items AS I
7 ON SL.ItemCode=I.ItemCode
8 GROUP BY SP.SalesPersonName
9 HAVING COUNT(DISTINCT SL.ItemCode)=(SELECT DISTINCT COUNT(ItemCode) FROM Items)
10
11 ]
```

150 % No issues found Ln: 11, Ch: 1 TABS

Results Messages

	SalesPersonName
1	דני

4. כתבו שאילתה המציגה את הפריטים שנמכרו הן אצל איש המכירות שמכר את המגוון הגדול ביותר בקטלוג והן אצל איש המכירות שמכר את הכמות הגדולה ביותר - אך לא נמכרו אצל איש המכירות שמכר את המגוון הקטן ביותר בקטלוג.

```

1  WITH MaxQty AS (
2      SELECT TOP 1 SalesPersonCode
3      FROM SalesHeader SH
4      JOIN SalesLine SL ON SH.DocNum = SL.DocNum
5      GROUP BY SH.SalesPersonCode
6      ORDER BY SUM( SL.Qty) DESC
7  ),
8  MaxVariety AS (
9      SELECT TOP 1 SalesPersonCode
10     FROM SalesHeader SH
11     JOIN SalesLine SL ON SH.DocNum = SL.DocNum
12     GROUP BY SH.SalesPersonCode
13     ORDER BY COUNT(DISTINCT SL.ItemCode) DESC
14 ),
15  MinVariety AS (
16     SELECT TOP 1 SalesPersonCode
17     FROM SalesHeader SH
18     JOIN SalesLine SL ON SH.DocNum = SL.DocNum
19     GROUP BY SH.SalesPersonCode
20     ORDER BY COUNT(DISTINCT SL.ItemCode) ASC
21 )
22  SELECT ItemCode FROM SalesLine SL
23  JOIN SalesHeader AS SH
24  ON SH.DocNum = SL.DocNum
25  WHERE SH.SalesPersonCode IN (SELECT SalesPersonCode FROM MaxQty)
26  INTERSECT
27  SELECT ItemCode FROM SalesLine SL
28  JOIN SalesHeader AS SH
29  ON SH.DocNum = SL.DocNum
30  WHERE SH.SalesPersonCode IN (SELECT SalesPersonCode FROM MaxVariety)
31  EXCEPT
32  SELECT ItemCode FROM SalesLine SL
33  JOIN SalesHeader AS SH
34  ON SH.DocNum = SL.DocNum
35  WHERE SH.SalesPersonCode IN (SELECT SalesPersonCode FROM MinVariety);
36
37

```

70 %



No issues found

Results

Messages

	ItemCode
1	1001
2	1002
3	1003
4	3611010

5. עבור כל איש מכירות מצאו את הפריטים שנמכרו מתחת לממוצע מכירות לפריט של אותו איש מכירות, כתבו שאילתה המציגה את איש המכירות, מספרי הפריטים ממוצע מכירות ליחידה ואת סכום המכירות.

SQLQuery (.....54) (שמות)

```
1 WITH ItemAvg AS (  
2     SELECT SH.SalesPersonCode, SL.ItemCode, AVG(SL.Qty) AS AvgQty  
3     FROM SalesHeader SH  
4     JOIN SalesLine SL ON SH.DocNum = SL.DocNum  
5     GROUP BY SH.SalesPersonCode, SL.ItemCode  
6 )  
7 SELECT  
8     SP.SalesPersonName, SL.ItemCode, IA.AvgQty, SL.Qty, SL.LineSum  
9     FROM SalesLine SL  
10    JOIN SalesHeader SH ON SH.DocNum = SL.DocNum  
11    JOIN SalesPerson SP ON SH.SalesPersonCode = SP.SalesPersonCode  
12    JOIN ItemAvg IA ON  
13        IA.SalesPersonCode = SH.SalesPersonCode  
14        AND IA.ItemCode = SL.ItemCode  
15    WHERE SL.Qty < IA.AvgQty  
16
```

100 % No issues found

Results Messages

	SalesPersonName	ItemCode	AvgQty	Qty	LineSum
1	דני	1003	2	1	70.00

6. כתבו שאילתה המציגה את אנשי המכירות שתרומתם מהווה 88% מסך הכמות הנמכרת. עבור כל איש מכירות, הציגו את הכמות הנמכרת בחלוקה לפי מספרי חשבוניות, כך שאנשי המכירות ימוינו לפי הכמות הנמכרת בסדר יורד, ובתוך כל איש מכירות החשבוניות ימוינו לפי תאריך יצירתן.

SQLQuery2....77) * (ששתמש))

SQLQuery1.s...54) * (ששתמש))

```

1  WITH SalesData AS (
2      SELECT
3          SH.SalesPersonCode, SL.DocNum, SH.DocDate, SUM(SL.Qty) AS Qty
4      FROM SalesHeader SH
5      JOIN SalesLine SL ON SH.DocNum = SL.DocNum
6      GROUP BY SH.SalesPersonCode, SL.DocNum, SH.DocDate
7  ),
8  SalesTotal AS (
9      SELECT SalesPersonCode, SUM(Qty) AS TotalQty
10     FROM SalesData
11     GROUP BY SalesPersonCode
12 ),
13 Ranked AS (
14     SELECT
15         ST.SalesPersonCode, SP.SalesPersonName, SD.DocNum, SD.DocDate, SD.Qty, ST.TotalQty,
16         SUM(ST.TotalQty) OVER (ORDER BY ST.TotalQty DESC
17                                 ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
18         * 1.0 / SUM(ST.TotalQty) OVER () AS CumulativePercent
19     FROM SalesTotal ST
20     JOIN SalesData SD ON ST.SalesPersonCode = SD.SalesPersonCode
21     JOIN SalesPerson SP ON ST.SalesPersonCode = SP.SalesPersonCode
22 )
23 SELECT
24     SalesPersonName, DocNum, DocDate, Qty
25 FROM Ranked
26 WHERE CumulativePercent <= 0.88
27 ORDER BY TotalQty DESC, SalesPersonCode, DocDate
28

```

91 %

No issues found

Results

Messages

	SalesPersonName	DocNum	DocDate	Qty
1	דני	101	2023-01-01	3
2	דני	102	2023-01-02	4
3	דני	105	2023-01-10	5

7. כתבו שאילתה המציגה עבור כל איש מכירות (SalesPersonCode) את סכום המכירות הכולל שלו, וכן את ממוצע סכום המכירות לכל חשבונית שלו. (יש לממש זאת באמצעות תת שאילתה ב-SELECT).

SQLQuery3....51))*(שנתמש) * SQLQuery2.s...77))*(חשתמש) * SQLQuery1.s....5

```

1  SELECT
2      SH.SalesPersonCode,
3      SUM(SL.LineSum) AS TotalSales,
4      (
5          SELECT AVG(InvoiceTotal)
6          FROM (
7              SELECT SUM(SL2.LineSum) AS InvoiceTotal
8              FROM SalesHeader SH2
9              JOIN SalesLine SL2 ON SH2.DocNum = SL2.DocNum
10             WHERE SH2.SalesPersonCode = SH.SalesPersonCode
11             GROUP BY SH2.DocNum
12         ) AS Invoices
13     ) AS AvgInvoiceAmount
14 FROM SalesHeader SH
15 JOIN SalesLine SL ON SH.DocNum = SL.DocNum
16 GROUP BY SH.SalesPersonCode
17

```

100 % No issues found

Results Messages

	SalesPersonCode	TotalSales	AvgInvoiceAmount
1	1	710.00	236.666666
2	2	260.00	260.000000
3	3	70.00	70.000000

חלק ב

בעיה א - זיהוי משתמשים תובעניים

משתמש תובעני זהו משתמש ששלח למערכת יותר מ-10 בקשות במהלך חלון זמן (כלשהוא) של 5 דקות.

יש לממש שאילתה המציגה את המשתמשים/ים התובעניים ביותר.

The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are several tabs for different queries. The active query is 'SQLQuery4....52) *'. The query text is as follows:

```
1 WITH WindowCounts AS (  
2     SELECT  
3         A.UserID,  
4         A.RequestTime AS WindowStart,  
5         COUNT(*) AS RequestsInWindow  
6     FROM AccessRequests A  
7     JOIN AccessRequests B  
8     ON A.UserID = B.UserID  
9     AND B.RequestTime BETWEEN A.RequestTime AND DATEADD(MINUTE, 5, A.RequestTime)  
10    GROUP BY A.UserID, A.RequestTime  
11 )  
12 SELECT UserID, MAX(RequestsInWindow) AS MaxRequests  
13 FROM WindowCounts  
14 GROUP BY UserID  
15 HAVING MAX(RequestsInWindow) > 10  
16 ORDER BY MaxRequests DESC  
17
```

Below the query window, there is a toolbar with a zoom level of 100%, 13 errors, and 0 warnings. Below the toolbar, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is active, showing a table with the following data:

	UserID	MaxRequests
1	101	16

הפתרון:

דבר ראשון הבנת העקרון שנדרש לעבוד בשאלה זו עם "חלון זז"- כלומר להתקדם כל הזמן עם עוגן משתנה בזמן קבוע – לבדוק בכל משתמש על כל בקשה החל ממנה ועד פרק זמן של 5 דקות. כך לעבור על כל בקשה בנפרד

לכל בקשה יש UserID ו RequestTime אלו העמודות הקריטיות לשאלה זו

שלב ראשון - # עבודה עם JOIN SELF לעבור כביכול בלולאה פנימית על הטבלה כדי למצוא לכל בקשה- החל ממנה עוד טווח של בקשות, בעצם לכל בקשה A לזהות את כל בקשות B בטווח של 5 דקות .

קיבוץ לפי ה UserID ו RequestTime של A
ספירה כמות של הבקשות שנופלות בטווח של 5 דקות קדימה

שלב שני – על בסיס תוצאת שלב 1
בדיקה לכל משתמש בנפרד - מתוך כלל החלונות שחושבו את כמות הבקשות המקסימלית שהיו לו בתוך חלון אחד
החזרה של המשתמשים שעמדו בתנאי-יותר מ 10 בקשות

בעיה ב - מקסום טיפול בבקשות דחופות

בהנחה שהמערכת יכולה לטפל בבקשה אחת בלבד בכל רגע נתון,

יש לממש שאילתה המחזירה את רצף הבקשות הכדאית לטיפול כך שסכום רמות העדיפות (Priority) של כלל

הבקשות שיטופלו יהיה מקסימלי.

יש לשמור על מגבלת הזמן של הבקשות (ExpirationTime) וכן לשמור על סדר שליחת הבקשות (RequestTime).

The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are tabs for 'SQLQuery7....(74) (ששתמש)*', 'SQLQuery6.s....(76) (משתמש)*', 'SQLQuery5.s....(55) (משתמש)*', and 'SQLQuery'. The active window is 'SQLQuery7....(74) (ששתמש)*'. It contains a T-SQL query that uses a CTE named 'Ordered' to rank requests by priority and request time, and another CTE named 'GreedyPick' to select requests based on these criteria. The query is as follows:

```
1 WITH Ordered AS (  
2     SELECT *,  
3     ROW_NUMBER() OVER (ORDER BY Priority DESC, RequestTime) AS RowNum  
4     FROM AccessRequests  
5 ),  
6 GreedyPick AS (  
7     SELECT A.*  
8     FROM Ordered A  
9     WHERE NOT EXISTS (  
10        SELECT 1  
11        FROM Ordered B  
12        WHERE B.RowNum < A.RowNum  
13            AND B.ExpirationTime > A.RequestTime  
14            AND A.RequestTime >= B.RequestTime  
15    )  
16 )  
17 SELECT *  
18 FROM GreedyPick  
19 ORDER BY RequestTime;  
20
```

Below the query window, there is a toolbar with a zoom level of 100%, a status bar showing 6 errors and 0 warnings, and a 'Results' tab. The 'Results' tab is active, displaying a table with 9 columns: RequestID, UserID, DocumentID, RequestTime, ResponseTime, Priority, ExpirationTime, and RowNum. The table contains 3 rows of data:

	RequestID	UserID	DocumentID	RequestTime	ResponseTime	Priority	ExpirationTime	RowNum
1	1	101	501	2025-12-28 10:00:00.000	2025-12-28 10:00:10.000	3	2025-12-28 11:00:00.000	3
2	2	101	502	2025-12-28 10:01:00.000	2025-12-28 10:01:05.000	4	2025-12-28 11:00:00.000	1
3	12	102	601	2025-12-28 11:00:00.000	2025-12-28 11:00:10.000	3	2025-12-28 12:00:00.000	8

הפתרון:

המטרה שלי זה למצוא רצף חוקי של בקשות ככה שיתקבל סכום רמות ה Priority המקסימלי אבל יחד עם זאת לשמור על התנאים:

#בקשה אחת בלבד בכל זמן

#שמירה על ExpirationTime

#שמירה על סדר השליחה (RequestTime)

זהו פתרון גרידי: בכל שלב נבחר את הבקשה הכי טובה שנכנסת חוקית.

ולכן שלבי הפתרון הם:

מיון - סידור כלל הבקשות לפי סדר עדיפויות (Priority DESC), ורק אחריו RequestTime כדי לשמור יציבות.

מספור - שימוש ב־ ROW_NUMBER() לכל בקשה לצורך שליטה בסדר הבחירה.

בחינה - לכל בקשה נבדקת אפשרות החפיפה שלה עם בקשות קודמות שכבר נבחרו לפי הקריטריון: אותה בקשה לא תיבחר אם יש קודמת לה שעדיין לא פג תוקפה

(ExpirationTime > RequestTime)

בחירה - רק בקשות חוקיות, לא חופפות, נשמרות לתוצאה הסופית.

החזרה - תוצאת השאילתה כוללת רק את הבקשות שבפועל ייכנסו לרצף הטיפול.

הערת הגשה:

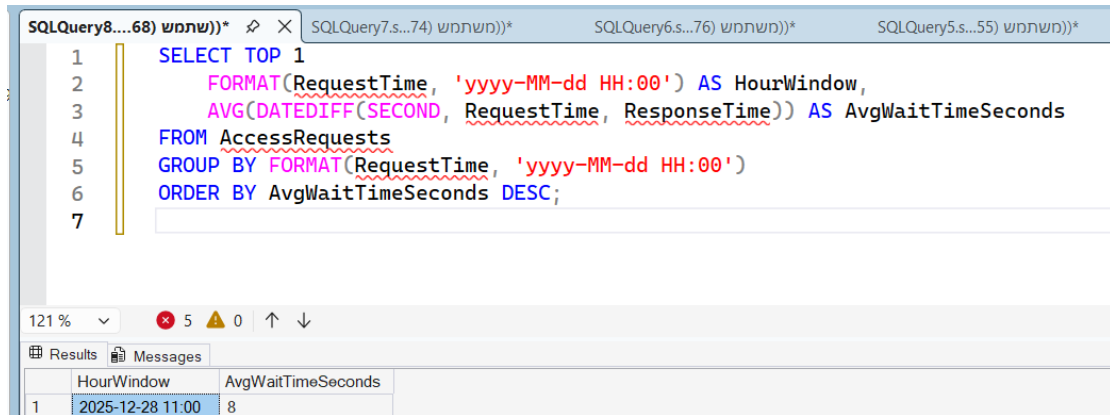
הפתרון אינו בהכרח אופטימלי – ייתכן שקיימת קבוצה אחרת חוקית, עם סכום עדיפויות

גבוה יותר, אך SQL לא תומכת בחישוב רצף דינמי כזה.

לכן בחרתי גישה **גרידית** – **מקורבת**, שהיא פתרון חוקי, יעיל, ומקובל בתנאי השפה.

בעיה ג - זיהוי צווארי בקבוק

יש לממש שאילתה המציגה את פרק הזמן שבו זמן ההמתנה הממוצע בין שליחת הבקשה לטיפול היה הגבוה ביותר.



```
1 SELECT TOP 1
2     FORMAT(RequestTime, 'yyyy-MM-dd HH:00') AS HourWindow,
3     AVG(DATEDIFF(SECOND, RequestTime, ResponseTime)) AS AvgWaitTimeSeconds
4 FROM AccessRequests
5 GROUP BY FORMAT(RequestTime, 'yyyy-MM-dd HH:00')
6 ORDER BY AvgWaitTimeSeconds DESC;
7
```

121 % 5 0

HourWindow	AvgWaitTimeSeconds
2025-12-28 11:00	8

הפתרון:

המטרה היא לזהות את פרק הזמן שבו זמן ההמתנה הממוצע של בקשות שטופלו היה הגבוה ביותר. לצורך כך נדרש לבצע עיבוד על זמני הבקשות, לקבץ אותם לפי חלונות זמן, ואז למצוא את הקבוצה עם ממוצע ההמתנה הגבוה ביותר

שלבי הפתרון:

חישוב - חישוב זמן ההמתנה של כל בקשה DATEDIFF – בין RequestTime ל-ResponseTime.

קיבוץ - שיוך כל בקשה לחלון זמן (למשל שעה עגולה) באמצעות FORMAT, ולאחר מכן קיבוץ לפי חלון זה וחישוב ממוצע זמן ההמתנה. (AVG)

סינון - מיון לפי ממוצע יורד (ORDER BY ... DESC) ובחירת השעה שבה הממוצע הוא הגבוה ביותר. (TOP 1)

חלק ג

סעיף א:

יש לכתוב שאילתה המחזירה את כל הקומבינציות האפשריות של שלושה ערכים שונים מתוך הטבלה עם חשיבות לסדר (כל סידור ייחשב כתוצאה נפרדת).

89

```
1 DECLARE @X INT = 32;
2 SELECT
3     Numbers1.val AS val1,
4     Numbers2.val AS val2,
5     Numbers3.val AS val3
6 FROM Numbers Numbers1
7 JOIN Numbers Numbers2 ON Numbers1.val <> Numbers2.val
8 JOIN Numbers Numbers3 ON Numbers1.val <> Numbers3.val
9                        AND Numbers2.val <> Numbers3.val
10 WHERE Numbers1.val + Numbers2.val + Numbers3.val = @X;
11
12
13
```

121 % 15 0 ↑ ↓

Results Messages

	val1	val2	val3
1	15	-5	22
2	15	22	-5
3	15	17	0
4	15	0	17
5	-7	17	22
6	-7	22	17
7	22	-5	15
8	22	17	-7
9	22	15	-5
10	22	-7	17
11	-5	22	15
12	-5	15	22
13	0	17	15
14	0	15	17
15	17	0	15
16	17	22	-7
17	17	-7	22
18	17	15	0

סעיף ב:

יש לכתוב שאילתה המחזירה את כל הקומבינציות האפשריות של שלושה ערכים שונים מתוך הטבלה ללא חשיבות לסדר (כל קומבינציה תופיע פעם אחת בלבד, ללא קשר לסדר הערכים בה).

```

1  DECLARE @X INT = 32;
2
3  SELECT
4      Numbers1.val AS val1,
5      Numbers2.val AS val2,
6      Numbers3.val AS val3
7  FROM Numbers AS Numbers1
8  JOIN Numbers AS Numbers2 ON Numbers1.val > Numbers2.val
9  JOIN Numbers AS Numbers3 ON Numbers1.val > Numbers3.val
10     AND Numbers2.val > Numbers3.val
11  WHERE Numbers1.val + Numbers2.val + Numbers3.val = @X;
12
13

```

121 % 15 0 ↑ ↓

Results Messages

	val1	val2	val3
1	22	17	-7
2	22	15	-5
3	17	15	0

סעיף ג:

נניח שפלט השאילתה של סעיף ב' נשמר בטבלה זמנית #temp_table.

יש לכתוב שאילתה שמחזירה את הקומבינציה שהמכפלה של שלושת הערכים בה היא הגדולה ביותר מבין כל הקומבינציות בטבלה הזמנית.

```

1  DECLARE @X INT = 32;
2
3  with maxK AS
4  (SELECT
5      Numbers1.val AS val1,
6      Numbers2.val AS val2,
7      Numbers3.val AS val3
8  FROM Numbers AS Numbers1
9  JOIN Numbers AS Numbers2 ON Numbers1.val > Numbers2.val
10  JOIN Numbers AS Numbers3 ON Numbers1.val > Numbers3.val
11     AND Numbers2.val > Numbers3.val
12  WHERE Numbers1.val + Numbers2.val + Numbers3.val = @X
13  )
14
15  SELECT TOP (1)*,
16      val1 * val2 * val3 AS product
17  FROM maxK
18  ORDER BY product DESC;
19

```

column product(, null)

100 % 15 0 ↑ ↓

Results Messages

	val1	val2	val3	product
1	17	15	0	0

חלק ד

יש לממש באופן עצמאי פונקציה ב-SQL שמקבלת מחרוזת והופכת אותה, בדומה לפונקציה המובנית REVERSE, אך ללא שימוש בפונקציה המקורית.
נדרש להראות שימוש בפונקציה החדשה בתוך שאילתת SELECT.

SQLQuery4....71) * (ששתמש) X SQLQuery3.s...77) * (משתמש) SQLQuery2.s...53) * (משתמש)

```
1 DECLARE @txt VARCHAR(100) = 'logicel';
2
3 WITH ReverseCTE AS (
4     SELECT
5         CAST(' ' AS VARCHAR(MAX)) AS reversed,
6         LEN(@txt) AS i
7     UNION ALL
8     SELECT
9         reversed + SUBSTRING(@txt, i, 1),
10        i - 1
11    FROM ReverseCTE
12    WHERE i > 0
13 )
14 SELECT TOP (1) @txt AS original, reversed AS reversed_result
15 FROM ReverseCTE
16 WHERE i = 0;
```

110 % 6 0

Results Messages

	original	reversed_result
1	logicel	lecigol

חלק ה

מטרה: להוסיף עמודה חדשה (KOTERET) שבה לכל ערך בעמודה SHURA מוקצה הערך הראשון של כל קבוצה

מתוך העמודה TEUR.

זדר פעולות:

- מיון הרשומות לפי SHURA.
- זיהוי התחלת קבוצה חדשה: שורה ראשונה או שורה שבה הערך הקודם ב-TEUR הוא NULL.
- מריחת הערך הראשון של כל קבוצה בעמודה TEUR לכל השורות באותה קבוצה בעמודה החדשה KOTERET.

```
1 IF COL_LENGTH('vals', 'KOTERET') IS NULL
2 BEGIN
3     ALTER TABLE Valve
4     ADD KOTERET VARCHAR(100);
5 END;
6 WITH FirstValidTeur AS (
7     SELECT
8         SHURA,
9         TEUR,
10        ROW_NUMBER() OVER (PARTITION BY SHURA ORDER BY ID) AS rn
11     FROM vals
12     WHERE TEUR IS NOT NULL
13 ),
14 KoteretPerShura AS (
15     SELECT SHURA, TEUR AS KOTERET
16     FROM FirstValidTeur
17     WHERE rn = 1
18 )
19 UPDATE v
20 SET v.KOTERET = k.KOTERET
21 FROM vals v
22 LEFT JOIN KoteretPerShura k ON v.SHURA = k.SHURA;
23 SELECT * FROM vals;
24
```

110 % No issues found

Results		Messages		
	ID	SHURA	TEUR	KOTERET
1	1	A	NULL	אבטיח
2	2	A	אבטיח	אבטיח
3	3	A	בננה	אבטיח
4	4	B	NULL	מנגו
5	5	B	מנגו	מנגו
6	6	C	NULL	NULL
7	7	C	NULL	NULL