

תשובות

ולג

1. כתבו שאלתה המציגת את הכמות, סכום המכירות וכמות החשבוניות לכל פריט.

```

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

```

	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 שאלתה המציגת את מספרי החשבוניות שבכל אחת מהן קיים פריט 3611010 ו אף 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

```

	DocNum
1	103
2	105

3. כתבו שאלתה המזינה את אנשי המכירות שמכרו את כל הפריטים המוצעים בקטלוג.

```
SQLQuery5...77 (משתמש)* SQLQuery4...55 (משתמש)* SQLQuery3...71 (משתמש)* DESKTOP-J898... - dbo.Items SQLQuery2...59 (משתמש)*
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 טלי

4. כתבו שאלתה המציגת 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

	ItemCode
1	1001
2	1002
3	1003
4	3611010

5. עבור כל איש מכירות מצאו את הפריטים שנמכרו מתחילה ממוצע מכירות לפרט של אותו איש מכירות, כתבו שאליתה המציגת את איש המכירות, מספר הפריטים ממוצע מכירות יחידה ואת סכום המכירות.

```

SQLQuery  (((שונט 45....))
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

	SalesPersonName	ItemCode	AvgQty	Qty	LineSum
1	לני	1003	2	1	70.00

6. כתבו שאלתה המציגת את אנשי המכירות שתרומותם מhoeה 88% משך הכמות הנמכרת. עבור כל איש מכירות, הציגו את הכמות הנמכרת בחלוקת לפי מספרי חשבון, כך שאנשי המכירות ימוננו לפי הכמות הנמכרת בסדר יורד, ובתוך כל איש מכירות החשבונות ימוננו לפי תאריך יצירתן.

SQLQuery2....77(*))((השתמש (SQLQuery1.s...54(*))((משמש

```

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

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

7. כתבו שאלתה המציגת עבור כל איש מכירות (SalesPersonCode) את סכום המכירות הכלל שלו, וכן את ממוצע סכום המכירות לכל חשבונית שלו. (יש למשוך זאת באמצעות תת שאלתה ב-`SELECT`.)

SQLQuery3....51 (משתמש)* SQLQuery2.s...77 (משתמש)* SQLQuery1.s...

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

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

حل ב

בעיה א - זיהוי משתמשים טובעניים

משתמש טובעני זהו משתמש שלח למערכת יותר מ-10 בקשות במהלך חלון זמן (כלשהו) של 5 דקות.

יש למשש שאלתה המציגת את המשתמשים התובעניים ביותר.

The screenshot shows a SQL Server Management Studio interface with four tabs at the top: SQLQuery6.s...76, SQLQuery5.s...55, SQLQuery4....52, and SQLQuery3.s...51. The SQLQuery4....52 tab contains the following T-SQL code:

```
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
```

The Results tab displays the output of the query:

	UserID	MaxRequests
1	101	16

הפתרון:

דבר ראשון הבנת הערךן שנדרש לעבוד בשאלת זו עם "חלון 5"-כזכור להתקדם כל הזמן עם עוגן משתנה בזמן קבוע – לבדוק בכל משתמש על כל בקשה החל ממנו ועד פרק זמן של 5 דקות. כר לעבור על כל בקשה בנפרד

לכל בקשה יש UserID ו RequestTime או העמודות הקritisיות לשאלת זו

שלב ראשון - # עבדה עם SELF JOIN לעבור כביכול בלולאה פנימית על הטבלה כדי למצוא לכל בקשה- החל ממנו עד טווח של בקשות, בעצם לכל בקשה A לזרות את כל בקשה בטווח של 5 דקות .

קיבוץ לפי ה UserID ו RequestTime של A
ספירה כמהות של הבקשות שנoverlapו בטווח של 5 דקות קדימה

שלב שני – על בסיס תוצאות שלב 1 בדיקה לכל משתמש בנפרד -מתוך כלל החולנות שחושו את כמהות הבקשות המקסימלית שהיו לו בתוך חלון אחד החזירה של המשתמשים שעמדו בתנאי-יוטר מ 10 בקשות

בעיה ב - מקסום טיפול בבקשתות דוחפות

בנהנזה שהמערכת יכולה לטפל בבקשת אחת בלבד בכל רגע נתון,

יש למשן שאליתה המחזירה את רצף הבקשות הקדאית לטיפול כך שסכום רמות העדיפות (Priority) של כל

הבקשות שיוטפלו יהיה מаксימלי.

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

The screenshot shows a SQL Server Management Studio interface. The top bar has tabs for 'SQLQuery7...74' (selected), 'SQLQuery6.s...76', 'SQLQuery5.s...55', and 'SQLQuery'. The main area contains a T-SQL script:

```
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;
```

The results pane shows a table with the following 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

פתרונות:

המטרה שלי זה למצוא רצף חוקי של בקשות ככה שיתקבל סכום רמות ה Priority

המקסימלי אבל יחד עם זאת לשמר על התנאים:

#בקשה אחת בלבד בכל זמן

#שמירה על ExpirationTime

#שמירה על סדר השליחה (RequestTime)

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

ולכן שלבי הפתרון הם:

מיון - סידור כלל הבקשות לפי סדר עדיפויות (Priority DESC), ורך אחרי RequestTime כדי לשמר 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 הקритריון : אותה בקשה לא תיבחר אם יש קודמת לה שעדיין לא פג תוקפה

(ExpirationTime > RequestTime)

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

החזרה - תוצאה השאלתה כוללת רק את הבקשות שבפועל ייכנסו לרצף הטיפול.

הערות הגשה:

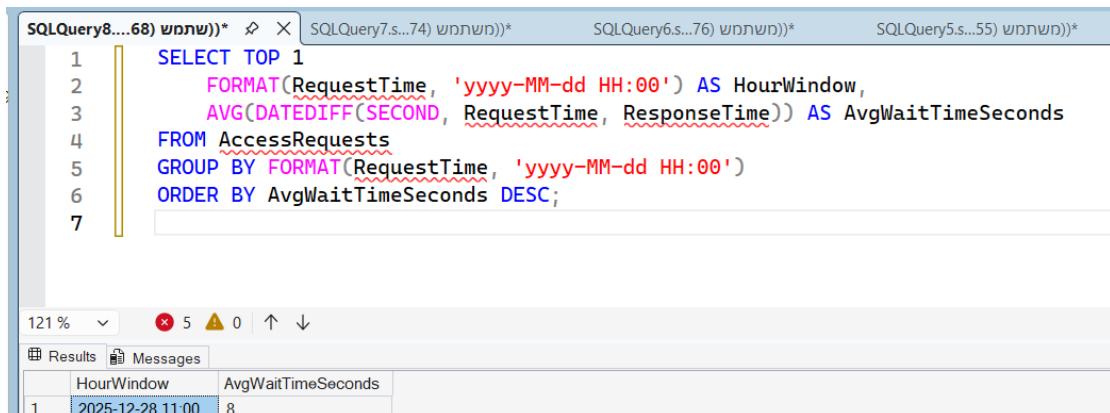
פתרון אינו בהכרח אופטימי – ייתכן שקיים מעת קבוצה אחרת חוקית, עם סכום עדיפויות

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

לכן בחרתי גישה גרידית – **מקורבת**, שהיא פתרון חוקי, עיל, ומתקבל בתנאי השפה.

בעיה ג - זיהוי צוואר בקברוק

יש למש שאליתה המציגת את פרק הזמן שבו זמן המתנה הממוצע בין שליחת הבקשה לטיפולה היה הגובה ביותר.



The screenshot shows a SQL Server Management Studio interface. In the top pane, there are four tabs: 'SQLQuery8....68' (selected), 'SQLQuery7.s...74', 'SQLQuery6.s...76', and 'SQLQuery5.s...55'. The query in the selected tab is:

```
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
```

In the bottom pane, the 'Results' tab is selected, showing the following table:

	HourWindow	AvgWaitTimeSeconds
1	2025-12-28 11:00	8

הפתרון:

המטרה היא לזיהות את פרק הזמן שבו זמן המתנה הממוצע של בקשות שטופלו היה הגבוה ביותר. לצורך כך נדרש לבצע עיבוד על זמני הבקשות, לפחות אוטם לפי חלונות זמן, ואז למצוא את הקבוצה עם ממוצע המתנה הגבוה ביותר.

שלבי הפתרון:

чисוב - חישוב זמן המתנה של כל בקשה DATEDIFF – בין RequestTime – לבין ResponseTime.

קיבוץ - שייר כל בקשה לחילון זמן (למשל שעה עגולה) באמצעות FORMAT, ולאחר מכן קיבוץ לפי חילון זה וчисוב ממוצע זמן המתנה.(AVG)

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

حلק ג

סעיף א:

יש לכתוב שאלתה המחזירה את כל הקומבינציות האפשרות של שלושה ערכים שונים מתוך הטבלה **עם חסיבות** לשדר (כל סידור יהיה כהוצאה נפרדת).

The screenshot shows a SQL query window with the following code:

```
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 Numbers1
8 JOIN Numbers Numbers2 ON Numbers1.val <> Numbers2.val
9 JOIN Numbers Numbers3 ON Numbers1.val <> Numbers3.val
10 AND Numbers2.val <> Numbers3.val
11 WHERE Numbers1.val + Numbers2.val + Numbers3.val = @X;
```

The results pane displays a table with three columns: val1, val2, and val3. The data is as follows:

	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 שאלתה המחזירה את כל הקומבינציות האפשריות של שלושה ערכים שונים מתוך הטבלה **לא חשיבות** לשדר (כל קומבינציה תופיע פעם אחת בלבד, ללא קשר לסדר הערך 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 השאלה של סעיף ב' נשמר בטבלה זמנית **#temp_table**.

יש לכתוב שאלתה שמחזירה את הקומבינציה שהמכפלה של שלושת הערכים בה היא הגדולה ביותר בין 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;
```

column product(, null)

100 % ✖ 15 ⚠ 0 ↑ ↓

Results Messages

	val1	val2	val3	product
1	17	15	0	0

ולק

יש למשמש באופן עצמאי פונקציה ב-SQL שמקבלת מחרוזת והופכת אותה, בדומה לפונקציה המובנית REVERSE.
נדרש להראות שימוש בפונקציה החדשה בתוך שאלתת SELECT.

```
SQLQuery4....71) (משתמש)* 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;
17
18
19
```

110 % ✖ 6 ⚠ 0 ↑ ↓

Results Messages

	original	reversed_result
1	logicel	leciogl

ולקח

צורה: להוסיף עמודה חדשה (KOTERET) שבה לכל ערך בעמודה SHURA מוקצת הערך הראשון של כל קבוצה מתוך העמודה TEUR.
אזור פעולות:

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

The screenshot shows the SQL Server Management Studio interface. The top bar has tabs for 'SQLQuery4.s...', 'SQLQuery3.s...', and 'SQLQuery4.s...'. The main area contains a script editor with the following T-SQL code:

```
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;
```

The status bar at the bottom left shows '110 %' and 'No issues found'. The bottom right pane shows the 'Results' tab with the following data:

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