קורס SQL למתחילים

הרצאה 5 – פתב"ס לתרגיל

חימום

חלק א'

כידוע, טבלת Posts מכילה פוסטים מסוגים שונים: שאלות, תשובות וכו'. העמודה PostTypeId מכילה מס' שמייצג את סוג הפוסט. הטבלה PostTypes מכילה מקרא של הסוגים. כתבו שאילתה שמביאה את 10,000 השורות הראשונות בטבלת Posts, ולכל אחד "מצמידה" את השם של הסוג שלו מטבלת PostTypes באמצעות JOIN.

פיתרון:

```
SELECT TOP 10000 Posts.*, PostTypes.Type
FROM Posts
JOIN PostTypes ON PostTypes.Id = Posts.PostTypeId
```

חלק ב'

3 ערכים בטבלת PostTypes הם כאלה שלא נעשה בהם שימוש בטבלת Posts. כתבו שמציפה את השורות הללו, תוך שימוש ב- JOIN בלבד ללא nested queries וללא שימוש ב- NOT IN, NOT EXISTS וכו'.

פיתרון:

```
SELECT PostTypes.Id,
            Type
  FROM [SO-2016].[dbo].[PostTypes]
  LEFT JOIN Posts ON Posts.PostTypeId = PostTypes.Id
 WHERE Posts.Id IS NULL
```

מה שאנחנו עושים בשאילתה הזאת זה לצרף לכל PostType את כל הפוסטים שנכתבו מאותו ה- Type. לצורך העניין, נוכל לדמיין שבתוצאה השורות של Question ושל Answers למשל משתכפלות מיליוני פעמים.
מכיוון שאנחנו עושים LEFT JOIN, אם יש PostType **שאין עבורו** פוסט, אז הערך של Posts.Id יהיה NULL (בניגוד ל- INNER JOIN שאז השורה הזאת פשוט לא הייתה קיימת). לכן, אנחנו מסתכלים **רק על אלה** שהערך הוא NULL, כלומר שאין עבורם פוסט מתאים.

תגים

כל שאלה ב- StackOverflow יכולה להיות משוייכת לתגיות שונות [מוקפות בעיגול שחור בתמונה].

שמות התגים נשמרים בטבלת Tags. לכל שורה יש Id של השורה (של התג) ו- TagName.
הקישור בין הפוסטים לבין התגיות מתבצע באמצעות טבלת PostsToTags שכוללת שתי עמודות: PostId (ה- Id של הפוסט, כמו שמופיע בטבלת Posts) ו- TagId (ה- Id של התג, כמו שמופיע בטבלת Tags).

כתבו שאילתה השולפת את כל השאלות שנכתבו ע"י משתמשים שה- Location שלהם הוא Israel ותוייגו בתג שמכיל את המילה 'sql'. העמודות שצריכות להיות בשליפה: ה-Id של השאלה, ה-Id של המשתמש שכתב את השאלה, שם המשתמש שכתב את השאלה, מספר השעות שעברו מיצירת השאלה למתן התשובה (רמז: ניתן להשתמש ב-DATEDIFF כמו בהרצאה) ושם המשתמש שענה, כותרת הפוסט ושם התג. את התוצאות מיינו לפי ה- Id של השאלה.

שימו לב שיש לכלול גם שורות של שאלות שלא נענו (להזכירכם, החלק של השאלות והתשובות דומה מאד לדוגמא מההרצאה).

פיתרון:

```
SELECT QuestionId = q.Id,
              HoursUntilAnswer = DATEDIFF(hour, q.CreationDate, qans.CreationDate),
              CreatorUserId = q.Id,
              CreatorUserDisplayName = quser.DisplayName,
              AnwerUserDisplayName = ansuser.DisplayName,
              PostTitle = q.Title,
              TagName = Tags.TagName
FROM Posts q
LEFT JOIN Posts qans ON qans.Id = q.AcceptedAnswerId
JOIN Users quser ON quser.Id = q.OwnerUserId AND quser.Location LIKE 'Israel'
LEFT JOIN Users ansuser ON ansuser.Id = qans.OwnerUserId
JOIN PostsToTags ON PostsToTags.PostId = q.Id
JOIN Tags ON Tags.Id = PostsToTags.TagId AND TagName LIKE '%sql%'
WHERE q.PostTypeId = 1
ORDER BY q.Id
```

מה שיש לנו פה זה פשוט חגיגה של JOIN-ים. שימו לב שאת התנאים השונים שנדרשים הכנסתי לתוך ה- ON ולא ב-WHERE. כמובן שהיה אפשר גם כך וגם כך.

משתמשים וקפיצות

טבלת המשתמשים (Users) כוללת Id רץ לכל משתמש. הרבה מה- Id-ים הם מספרים עוקבים 1,2,3,4,5. אבל, לעיתים יש "קפיצות". למשל, ה- Id שאחרי 5 הוא 8.

שלב ראשון:

כתבו שאילתה שמחזירה את המספרים שבהם התחילה קפיצה. למשל, שתי השורות הראשונות ייראו כך:

	JumpBegin
1	-1
2	5

הסיבה היא שב- -1 מתחילה קפיצה (אין משתמש ב-0, אלא המשתמש הבא הוא רק 1). בנוסף, גם ב- 5 מתחילה קפיצה (המשתמש שאחריו הוא רק ב- 8).

כתבו את השאילתה הזאת ללא שימוש ב- nested queries, אלא עם JOIN בלבד.

פיתרון:

```
SELECT JumpBegin = u1.Id
  FROM [SO-2016].[dbo].[Users] u1
  LEFT JOIN Users u2 ON u2.Id = u1.Id + 1
  where u2.Id is null
```

אנחנו הולכים פה לטבלה Users (כאשר את המקור אנחנו מכנים u1) ועושים Self Join (ומכנים את מה שאנחנו מצרפים u2) כאשר התנאי שלנו ל- JOIN הוא שאנחנו מנסים לצרף לכל שורה את המשתמש שה-Id שלו הוא המספר העוקב. אנחנו משתמשים ב- LEFT JOIN – כך שאלה שאין להם משתמש בעלי מספר עוקב, הערך ב- u2.Id יהיה NULL. ולכן, אנחנו מפלטרים על אלה שהערך שם הוא NULL כי אנחנו מתעניינים ספיציפית באלה שאין להם מס' עוקב.

שלב שני:

כתבו שאילתה שמחזירה שתי עמודות: ה- Id שבו התחילה הקפיצה, והגודל של הקפיצה. למשל, כך ייראו שתי השורות הראשונות:

	JumpBegin	JumpNum
1	-1	2
2	5	3

הסיבה היא שב- -1 יש משתמש, אבל אין משתמש ב-0 (המספר העוקב), אלא המשתמש הבא הוא ב-1. כלומר, קפיצה של 2. ב-5 יש משתמש, אבל אין ב-6 וב-7, אלא המשתמש הבא הוא ב-8. מכאן, קפיצה של 3.

כתבו שאילתה שמחזירה את הקפיצות שיש. בשלב הזה, ניתן לשלב JOIN ו- Nested Queries בשביל התוצאה הרצוייה.

פיתרון:

```
SELECT JumpBegin = u1.Id,
             JumpNum = (SELECT MIN(Id) FROM Users u3 WHERE u3.Id > u1.Id) - u1.Id
  FROM [SO-2016].[dbo].[Users] u1
  LEFT JOIN Users u2 ON u2.Id = u1.Id + 1
  where u2.Id is null
```

הפיתרון פה נשען על היתרון של חלק א', רק עם תוספת קטנה – אנחנו משתמשים ב- nested query כדי להביא את ה- Id המינימלי שגדול מה- Id שלנו, ומפחיתים ממנו את ה- Id שלנו כדי לקבל את גודל הקפיצה.