קורס SQL למתחילים

הרצאה 9 – תרגיל – פתב"ס

יצירת טבלאות

צרו 3 טבלאות באמצעות הממשק משתמש של SSMS, המייצגות פרטים של תחזית מזג אוויר.

<u>טבלת Forecaster</u>

מייצגת פרטים על חזאי, וכוללת 3 עמודות:

- ID (int, primary key, identity)
- FirstName ((nvarchar(50))
- LastName ((nvarchar(50))

<u>טבלת Cities</u>

מייצגת מידע על ערים שעבורם ניתנת תחזית. כוללת את העמודות הבאות:

- ID (int, primary key, identity)
- Name ((nvarchar(50))

<u>טבלת תחזית</u>

מייצגת מידע על תחזית שניתנה ליעד מסויים כוללת את העמודות הבאות:

- ID (int, primary key, identity)
- CityID (int, הגדירו relationship מול טבלת Cities)
- ForecasterID (int, הגדרו relationship מול טבלת Forecaster)
- DateGiven (datetime2, מייצגת את התאריך והשעה שבו ניתנה התחזית)
- IsRainy (bit, מייצג ערך בוליאני [0 או 1] של האם צפוי גשם)
- Temp (int, מכיל את הטמפרטורה)

גאווה לאומית

בחזרה ל- DB של StackOverflow. כתבו שאילתה שמעדכנת את שדה ה- AboutMe בטבלת Users. בצעו את העדכון עבור משתמשים שב- Location שלהם מופיע 'Israel' (השתמשו ב- LIKE), שה- AboutMe שלהם ריק (NULL או string ריק) ושענו **תשובה שהתקבלה** על לפחות 5 שאלות (כלומר, שהעמודה AcceptedAnswerId בלפחות 5 שאלות מצביעה על תשובות שנענו ע"י המשתמש) .
עבור משתמשים אלה, הכיתוב החדש ב- AboutMe צריך להיות 'Israeli Rockstar!'.

פיתרון:

```
with cte as (
	SELECT *
	FROM Users
	WHERE  Location LIKE '%Israel%' AND
			(AboutMe IS NULL OR AboutMe='') AND
			(SELECT COUNT(q.Id)
				FROM Posts q -- questions
				JOIN Posts a ON q.AcceptedAnswerId = a.Id --answers
				WHERE q.PostTypeId = 1 AND a.OwnerUserId=Users.Id
			) >=5

)
UPDATE cte
SET AboutMe = 'Israeli Rockstar'
```

אנחנו לוקחים ב- common table expression את המשתמשים מהטבלה Users, מפלטרים לפי ה- Location וה- AboutMe בצורה טריוויאלית. הקטע הטריקי היחיד (שגם מהווה קצת חזרה על חומר קודם בקורס) הוא להביא רק את אלה שענו על לפחות 5 שאלות. לכן אנחנו עושים Self-Join של הטבלה Posts על עצמה.
בתור q אנחנו מביאים רק את השאלות עצמן (זה מה שב- FROM, כשב- WHEREאנחנו מביאים רק פוסטים עם PostTypeId=1, כלומר שאלות – למרות שזה לא נחוץ, כי במילא רק לשאלות יהיה ערך בעמודה AcceptedAnswerId) ואנחנו עושים JOIN עם Posts שוב, הפעם תחת השם a, כדי לייצג את התשובות – כאשר אנחנו מצרפים לשאלה את התשובה שלה, ובגלל השימוש ב- JOIN (שזה קיצור ל- INNER JOIN) נקבל רק שורות שנמצאה עבורן התאמה – כלומר שאלות שנמצאו עבורן תשובות. אנחנו גם מפלטרים לפי שאלות שהמשתמש שענה עליהן הוא אותו המשתמש שאנחנו מסתכלים על השורה שלו עכשיו בטבלה Users.
כלומר, מספר השורות שנקבל יהיה מספר התשובות המשתמש הנ"ל ענה שהתקבלו – ואנחנו לוקחים רק את אלה שמעל 5.

סיכומי תגים

סעיף א'

כתבו CREATE TABLE statement שמתאר טבלה בשם TagsSummary שמכילה עבור כל תג מתי התפרסמה השאלה הראשונה והאחרונה באותו התג. הטבלה תכיל את העמודות הבאות (בסוגריים ה- type הדרוש):

- TagId (int)
- TagName (nvarchar(100))
- FirstPostTime (datetime)
- LastPostTime (datetime)

פיתרון:

```
CREATE TABLE TagsSummary
(
	TagId int not null,
	TagName nvarchar(100) not null,
	FirstPostTime datetime not null,
	LastPostTime datetime not null
)
```

סעיף ב'

כתבו שאילתת SELECT...INSERT שממלאת ערכים בטבלה הזאת עבור התאריכים 01/01/2016-01/04/2016.

פיתרון:

```
INSERT INTO TagsSummary(TagId, TagName, FirstPostTime, LastPostTime)
SELECT TagId, TagName, MIN(Posts.CreationDate), MAX(Posts.CreationDate)
FROM Posts
JOIN PostsToTags ON PostsToTags.PostId = Posts.Id
JOIN Tags ON Tags.Id = PostsToTags.TagId
WHERE Posts.PostTypeId=1 AND Posts.CreationDate BETWEEN '2016-01-01' AND '2016-04-01'
GROUP BY TagId, TagName
```

סעיף ג'

כתבו שאילתת MERGE שיכולה למזג לתוך הטבלה נתונים נוספים. מזגו לתוך הטבלה את כל הנתונים מה- 01/03/2016.

שימו לב: אמנם לא הודגש בהרצאה, אבל שאילתת MERGE חייבת להסתיים בנקודה-פסיק.

פיתרון:

```
MERGE TagsSummary AS target
USING  (
		SELECT TagId, TagName, MIN(Posts.CreationDate), MAX(Posts.CreationDate)
		FROM Posts
```

```
                    JOIN PostsToTags ON PostsToTags.PostId = Posts.Id
                    JOIN Tags ON Tags.Id = PostsToTags.TagId
                    WHERE Posts.PostTypeId=1 AND Posts.CreationDate > '2016-03-01'
                    GROUP BY TagId, TagName
                ) AS source (TagId, TagName, FirstSeen, LastSeen)
ON (target.TagId = source.TagId)
WHEN MATCHED AND source.LastSeen > target.LastPostTime OR source.FirstSeen < target.FirstPostTime THEN
       UPDATE SET    target.LastPostTime =       CASE WHEN source.LastSeen > target.LastPostTime
                                                       THEN source.LastSeen
                                                       ELSE target.LastPostTime
                                                  END,
                     target.FirstPostTime =      CASE WHEN source.FirstSeen < target.FirstPostTime
                                                       THEN source.FirstSeen
                                                       ELSE target.FirstPostTime
                                                  END
WHEN NOT MATCHED THEN
       INSERT (TagId, TagName, FirstPostTime, LastPostTime)
       VALUES (source.TagId, source.TagName, source.FirstSeen, source.LastSeen);
```