**מועד א – מילואים- 2025**

a.  שילפו את מספר ה tag ששמו 'Sql'

SELECT id

FROM tags

WHERE TagName = 'sql';

b.  שילפו את ה comments שה score שלהם הוא 100

\*ֵ SELECT

FROM comments

WHERE score = 100;

c. שילפו את ה comments שה text שלהם מכיל את המילה 'Sql'. יש לאתר את המילה

בכל case אך לא בתוך מילה כמו MySql. לצורך התרגיל ניתן להניח שמילה מוקפת

ברווחים, מתחילה טקסט או מסיימת אותו.

SELECT \*

FROM comments

WHERE LOWER(Text) LIKE '% sql %'

   OR LOWER(Text) LIKE 'sql %'

   OR LOWER(Text) LIKE '% sql'

   OR LOWER(Text) = 'sql';

d. בטבלת posts יש שדה tags. השדה מכיל רשימה של אינדקסי tags( לדוגמה ],4,7

82[(. שלפו את כל ה post שיש להם את ה tag מספר .7 שימו לב לא לשלוף את אלו

עם 17 או .70

SELECT \*

FROM posts

WHERE Tags LIKE '%,7,%'

   OR Tags LIKE '[7,%'

   OR Tags LIKE '%,7]'

   OR Tags = '[7]';

e. מיצאו את ה users ה"שנויים במחלוקת". אלו users שיש להם יותר מ 100 UpVotes

ו 100 DownVotes ושאחוז ה UpVotes מכלל ה votes שלהם הוא בין 45 ל .55

בונוס: שילפו, עבור ה users שעומדים במספר ה votes המינימלי את ממוצע ה

UpVotes לשנויים במחלוקת ולשאינם כאלה.

CREATE VIEW controversial\_users AS

SELECT \*

FROM users

WHERE UpVotes > 100

  AND DownVotes > 100

  AND (UpVotes \* 1.0) / (UpVotes + DownVotes) BETWEEN 0.45 AND 0.55;

SELECT \*

FROM controversial\_users;

**Bonusֵ :**

  SELECT

    CASE

      WHEN  (UpVotes \* 1.0) / (UpVotes + DownVotes) BETWEEN 0.45 AND 0.55

      THEN 'Controversial'

      ELSE 'Not Controversial'

    END AS controversy\_status,

    AVG(UpVotes) AS avg\_upvotes

  FROM users

WHERE UpVotes > 100

       AND DownVotes > 100

GROUP BY controversy\_status;

f. שילפו את כל ה post של OwnerUserId מספר ,7 מסודרים לפי CreationDate

SELECT \*

FROM posts

WHERE OwnerUserId = 7

ORDER BY CreationDate;

g. שילפו את כל ה votes של ה Posts של OwnerUserId מספר ,7 מסודרים לפי

CreationDate של ההצבעה בסדר יורד

SELECT v.PostId, v.VoteTypeId, v.CreationDate

FROM votes AS v

JOIN posts AS p ON v.PostId = p.Id

WHERE p.OwnerUserId = 7

ORDER BY v.CreationDate DESC;

h. שילפו את כל ה posts שה OwnerDisplayName שלהם אינו זהה לזה של ה user

שהוא ה OwnerUserId שלהם. בונוס: הכפילות בהחזקת OwnerDisplayName היא

פתח לבעיות. תנו נימוק אפשרי לבחירה להשתמש בכפילות.

SELECT p.Id, p.OwnerDisplayName, u.DisplayName

FROM posts AS p

JOIN users AS u ON p.OwnerUserId = u.Id

WHERE  p.OwnerDisplayName <> u.DisplayName;

למרות שכפילות בשדה פותחת פתח לחוסר עקביות, ייתכן שנבחר בה בשביל לשפר ביצועים, זהו צעד של   precompute להאצת זמן ריצה שמאפשר להאיץ חיפושים ולטעון רשימות פוסטים בלי חיבור לטבלת המשתמשים.  
אם המטרה היא שימור היסטוריית שמות, זו לא הדרך הנכונה -עדיף לממש טבלת היסטוריה ולשלוף את השם התקף במועד יצירת הפוסט.

i. שילפו את עשרת ה tags עם מספר ה count הגבוה ביותר. אין חשיבות לסדר הבחירה

במקרה של שוויון ב count

SELECT TagName, Count

FROM tags

ORDER BY Count DESC

LIMIT 10;

j. שילפו את מספר ה users

SELECT COUNT(\*) AS user\_count

FROM users;

k. שילפו את ה score הממוצע של post

SELECT AVG(Score) AS avg\_post\_score

FROM posts;

l. עבור כל score שילפו את מספר ה posts שקיבלו אותו', ממוינים לפי score

SELECT Score, COUNT(\*) AS post\_count

FROM posts

GROUP BY Score

ORDER BY Score;

m. שילפו את כל ה WebsiteUrl שהופיעו אצל יותר מ user יחיד. בונוס: האם כדאי למנוע

אפשרות זאת? אם כן, כיצד ניתן לממש זאת?

SELECT WebsiteUrl

FROM users

WHERE WebsiteUrl IS NOT NULL

GROUP BY WebsiteUrl

HAVING COUNT(\*) > 1;

**בונוס -**כן, כדאי למנוע כפילות ב־WebsiteUrl כאשר הכתובת מייצגת את המשתמש באופן ייחודי.  
 כפילות עלולה לגרום לבלבול ולפגיעה באמינות הנתונים לעומת ייחודיות המשפרת את איכות המידע ואת יעילות החיפוש. בשביל למנוע את הכפילות יש להוסיף מגבלת UNIQUE על העמודה.

ALTER TABLE users

ADD CONSTRAINT unique\_website UNIQUE (WebsiteUrl)ֵ;

n. שילפו את ה users שהם לא ה owners של אף post

SELECT u.\*

FROM users AS u

LEFT JOIN posts AS p ON u.Id = p.OwnerUserId

WHERE p.Id IS NULL;

o. שילפו את כל ה users שאף post שלהם לא קיבל אף vote

SELECT u.Id, u.DisplayName

FROM users u

JOIN posts p ON u.Id = p.OwnerUserId

LEFT JOIN (

SELECT DISTINCT p.OwnerUserId

FROM posts p

JOIN votes v ON p.Id = v.PostId

) AS voted\_users ON u.Id = voted\_users.OwnerUserId

WHERE voted\_users.OwnerUserId IS NULL;

p. שילפו את מספר ה comments לכל post, בעזרת טבלת comments

CREATE VIEW post\_comment\_counts AS

SELECT PostId, COUNT(\*) AS comment\_count

FROM comments

GROUP BY PostId;

SELECT \*

FROM post\_comment\_counts

q. עבור posts שקיבלו הערות, שילפו את ה distributions post per comments.

כלומר, מספר הפעמים ש post קיבל הערה ,1 ,2 3 וכדומה.

SELECT comment\_count, COUNT(\*) AS posts\_with\_that\_comment\_count

FROM post\_comment\_counts

GROUP BY comment\_count

ORDER BY comment\_count;

r. הרחיבו את הסעיף הקודם ל posts שלא קיבלו הערות.

CREATE VIEW post\_comment\_counts\_all AS

SELECT p.Id AS PostId, COUNT(c.Id) AS comment\_count

FROM posts p

LEFT JOIN comments c ON p.Id = c.PostId

GROUP BY p.Id;

SELECT comment\_count, COUNT(\*) AS posts\_with\_that\_comment\_count

FROM post\_comment\_counts\_all

GROUP BY comment\_count

ORDER BY comment\_count;

s. הציגו את זוגות ה ids post שיותר מ 10 אנשים זהים הצביעו לשניהם )כלומר אותו

אדם הצביע לשני ה posts). שילפו באופן סימטרי, כלומר )b,a )וגם )a,b )בונוס: האם

כדאי לשלוף זוגות רפלקסיבים )כלומר post עם עצמו(?

SELECT

  v1.PostId AS post\_a,

  v2.PostId AS post\_b,

  COUNT(DISTINCT v1.UserId) AS shared\_voters

FROM votes v1

JOIN votes v2

  ON v1.UserId = v2.UserId

WHERE v1.PostId <> v2.PostId         -- מסנן (a,a)

GROUP BY v1.PostId, v2.PostId

HAVING COUNT(DISTINCT v1.UserId) > 10;

**בונוס** - לא כדאי לשלוף זוגות רפלקסיבייםֵ, כי פוסט שמצביעים לו אין לו השפעה על עצמו והדבר לא תורם לניתוח של קשרים בין פוסטים.

t. בטבלה posts יש שדה בשם CommentsCount. בנוסף ניתן לחשב את השדה

בעזרת טבלת comments. כיתבו שאילתה המוצאת אי הסכמות בין השדה לטבלת

comments. השתמשו בשאילתה מסעיף קודם המוצאת את מספר ה comments

בעזרת טבלת comments

SELECT p.Id AS PostId, p.CommentsCount AS stored\_count, v.comment\_count AS actual\_count

FROM posts p

JOIN post\_comment\_counts AS v ON p.Id = v.PostId

WHERE p.CommentsCount <> v.comment\_count;