Data Integrity Q2 // Michal Buchbinder

מוטיבציה 1: עצלנות / חוסר זמן
סוג ההמלצה: העתקת סט המלצות קיים תחת משתמש/זהות אחרת.
ייראה בנתונים: שני משתמשים (או יותר) עם חפיפה כמעט מלאה של אותם זוגות סרטים.
השפעה על המערכת: הזוגות המשוכפלים מקבלים **משקל־יתר** במדדים (Precision/Recall וכו') ומטים את ההערכה, מה שייצור משקל רב יותר ולא מייצג עבור המלצות מסוימות.

נזהה את ההמלצות הללו בעזרת השליפה הבאה:

```
SELECT
  mr1.user_id AS u1,
  mr2.user_id AS u2,
  COUNT(*)     AS shared
FROM imdb_ijs.movies_recommendations mr1
JOIN imdb_ijs.movies_recommendations mr2
  ON mr1.user_id < mr2.user_id
 AND LEAST(mr1.movie1_id, mr1.movie2_id) = LEAST(mr2.movie1_id, mr2.movie2_id)
 AND GREATEST(mr1.movie1_id, mr1.movie2_id) = GREATEST(mr2.movie1_id,
mr2.movie2_id)
GROUP BY mr1.user_id, mr2.user_id
HAVING COUNT(*) >= 20;   -- סף לדוגמה
```

תיקון (בעזרת SQL):

```
DELETE FROM imdb_ijs.movies_recommendations mr
USING (
  SELECT
    mr1.user_id AS u1,
    mr2.user_id AS u2
  FROM imdb_ijs.movies_recommendations mr1
  JOIN imdb_ijs.movies_recommendations mr2
    ON mr1.user_id < mr2.user_id
   AND LEAST(mr1.movie1_id, mr1.movie2_id) = LEAST(mr2.movie1_id, mr2.movie2_id)
   AND GREATEST(mr1.movie1_id, mr1.movie2_id) = GREATEST(mr2.movie1_id,
mr2.movie2_id)
  GROUP BY mr1.user_id, mr2.user_id
  HAVING COUNT(*) >= 20
) dup_users
JOIN imdb_ijs.movies_recommendations base
  ON base.user_id = dup_users.u1
```

```
-- התאמה על אותו זוג סרטים מנורמל
 AND LEAST(base.movie1_id, base.movie2_id) = LEAST(mr.movie1_id, mr.movie2_id)
 AND GREATEST(base.movie1_id, base.movie2_id) = GREATEST(mr.movie1_id,
mr.movie2_id)
WHERE mr.user_id = dup_users.u2;     -- מוחקים רק אצל המעתיק
```

מוטיבציה 2: זדון (שיבוש/הטייה מכוונת)
סוג ההמלצה: חיבור ז'אנרים מנוגדים/קהל יעד לא תואם כדי לפגוע בהתאמה.
ייראה בנתונים: הרבה זוגות החוצים טבלת "ניגודי ז'אנרים" (למשל Action↔Romance, Comedy↔Horror).
ההשפעה על המודל: המודל שלנו ישען וילמד מתוך דאטא שקרי, וכך הוא ייצור קשרים שגויים ולא מהימנים. כתוצאה מכך, איכות ההמלצות הכללית שלו תרד, מאחר ואין דבר המבדיל בין ההמלצות האיכותיות להמלצות שאינן איכותיות, ובעיני המודל הן שוות ערך.

נזהה את ההמלצות הללו בעזרת השליפות הבאות:

```
WITH opposite_genres AS (
  SELECT 'Comedy' g1,'Horror' g2 UNION ALL
  SELECT 'Romance','Action' UNION ALL
  SELECT 'Family','Thriller' UNION ALL
  SELECT 'Animation','Crime'
)
SELECT DISTINCT mr.*
FROM imdb_ijs.movies_recommendations mr
JOIN imdb_ijs.movies_genres g1 ON g1.movie_id = mr.movie1_id
JOIN imdb_ijs.movies_genres g2 ON g2.movie_id = mr.movie2_id
JOIN opposite_genres og
  ON (LOWER(g1.genre)=LOWER(og.g1) AND LOWER(g2.genre)=LOWER(og.g2))
  OR (LOWER(g1.genre)=LOWER(og.g2) AND LOWER(g2.genre)=LOWER(og.g1));
```

תיקון (בעזרת SQL):

```
DELETE FROM imdb_ijs.movies_recommendations AS mr
USING (
  SELECT DISTINCT mr2.movie1_id AS movie_1, mr2.movie2_id AS movie_2
  FROM imdb_ijs.movies_recommendations mr2
  JOIN imdb_ijs.movies_genres g1 ON g1.movie_id = mr2.movie1_id
  JOIN imdb_ijs.movies_genres g2 ON g2.movie_id = mr2.movie2_id
  JOIN (
    SELECT 'Comedy' g1,'Horror' g2 UNION ALL
```

```
    SELECT 'Romance','Action'     UNION ALL
    SELECT 'Family','Thriller'    UNION ALL
    SELECT 'Animation','Crime'
  ) AS og
    ON (LOWER(g1.genre)=LOWER(og.g1) AND LOWER(g2.genre)=LOWER(og.g2))
     OR (LOWER(g1.genre)=LOWER(og.g2) AND LOWER(g2.genre)=LOWER(og.g1))
) AS bad
WHERE mr.movie1_id = bad.movie_1
  AND mr.movie2_id = bad.movie_2;
```

מוטיבציה 3: האיש הפלאי בחר להכניס המלצות על סרטים שהוא כנראה לא באמת מכיר, אלא פשוט לקח מהשורות הראשונות של טבלת הסרטים (IDs נמוכים, שנים ישנות מאוד). אלו סרטים נדירים שאף אחד אחר לא התייחס אליהם במערכת, ולכן ברור שהבחירה בהם לא נבעה מהיכרות אמיתית אלא מחיפוש קצר או חוסר השקעה.

סוגי ההמלצות: בחירה בסרטים עם מזהי ID ראשוניים ושנות הפקה מוקדמות מאוד, או סרטים שלא הופיעו כלל בהמלצות של משתמשים אחרים.

כיצד ייראו ההמלצות בנתונים:
זוגות סרטים עם movie_id נמוך במיוחד (למשל 1–100), או עם year ישן מאוד (לפני 1920), ושניהם אינם מופיעים בשום המלצה אחרת פרט לאותו משתמש. במערכת ייראו כ״יתומים״ – כלומר סרטים שאינם חוזרים כלל אצל יתר המשתמשים.

ההשפעה על המודל:
המודל ייחשף להמלצות שאין להן שום תוקף מציאותי או חפיפה עם התנהגות של משתמשים אחרים. כתוצאה מכך, הוא עלול לייחס חשיבות מופרזת לסרטים נדירים ובלתי רלוונטיים, לפגוע ב-Precision של המודל, ולבזבז זמן למידה על קשרים שאינם מייצגים העדפות אמיתיות של הצופים.

נזהה סרטים שמופיעים בהמלצות של משתמש יחיד, ואז מאתרים את כל ההמלצות שמבוססות על סרטים כאלה (אפשר לדרוש ששני צדי הזוג יתומים, או לפחות צד אחד).

```
WITH movie_users AS (
  SELECT movie1_id AS movie_id, user_id FROM imdb_ijs.movies_recommendations
  UNION ALL
  SELECT movie2_id AS movie_id, user_id FROM imdb_ijs.movies_recommendations
),
movie_usage AS (
  SELECT
    movie_id,
    COUNT(DISTINCT user_id) AS users_count,
    MIN(user_id) AS only_user_min,
```

```
    MAX(user_id) AS only_user_max
  FROM movie_users
  GROUP BY movie_id
)
המלצות שבהן שני הסרטים "יתומים" והיחיד שנגע בהם הוא אותו המשתמש שיצר את ההמלצה --
SELECT mr.*
FROM imdb_ijs.movies_recommendations mr
JOIN movie_usage mu1 ON mu1.movie_id = mr.movie1_id
JOIN movie_usage mu2 ON mu2.movie_id = mr.movie2_id
WHERE
  mu1.users_count = 1 AND mu2.users_count = 1
  AND mu1.only_user_min = mu1.only_user_max      -- נוודא שמשתמש יחיד
  AND mu2.only_user_min = mu2.only_user_max
  AND mr.user_id = mu1.only_user_min             -- ושהוא גם היוצר של ההמלצה
  AND mr.user_id = mu2.only_user_min;
```

ניתן לזהות גם לפי סרטים מתחילת הטבלה/שנים מוקדמות מאוד. אפשר לשלב שני תנאים:

- LOW_ID_THRESHOLD לדוגמה: 100
- OLD_YEAR_THRESHOLD לדוגמה: 1920

```
WITH movie_users AS (

  SELECT movie1_id AS movie_id, user_id FROM imdb_ijs.movies_recommendations

  UNION ALL

  SELECT movie2_id AS movie_id, user_id FROM imdb_ijs.movies_recommendations

),

movie_usage AS (

  SELECT

    movie_id,

    COUNT(DISTINCT user_id) AS users_count,

    MIN(user_id) AS only_user_min,

    MAX(user_id) AS only_user_max

  FROM movie_users
```

```
  GROUP BY movie_id
)
SELECT mr.*
FROM imdb_ijs.movies_recommendations mr
JOIN imdb_ijs.movies m1 ON m1.id = mr.movie1_id
JOIN imdb_ijs.movies m2 ON m2.id = mr.movie2_id
JOIN movie_usage mu1 ON mu1.movie_id = mr.movie1_id
JOIN movie_usage mu2 ON mu2.movie_id = mr.movie2_id
WHERE
  -- לפחות אחד מהסרטים "ראשון/ישן מאוד"
  (
    m1.id <= 100 OR m2.id <= 100
    OR m1.year < 1920 OR m2.year < 1920
  )
  -- שני הסרטים בשימוש של משתמש יחיד, והוא היוצר של ההמלצה
  AND mu1.users_count = 1 AND mu2.users_count = 1
  AND mu1.only_user_min = mu1.only_user_max
  AND mu2.only_user_min = mu2.only_user_max
  AND mr.user_id = mu1.only_user_min
  AND mr.user_id = mu2.only_user_min;
```

אופן המחיקה/התיקון של ההמלצות ממוטיבציה זו:

```
DELETE FROM imdb_ijs.movies_recommendations mr
USING (
  SELECT x.movie1_id, x.movie2_id, x.user_id
  FROM ( /* SELECT של השליפה למעלה */ ) AS table_above
) AS bad
WHERE mr.movie1_id = bad.movie1_id
  AND mr.movie2_id = bad.movie2_id
  AND mr.user_id   = bad.user_id;
```