2) תקינות נתונים

(1

כאשר מחשבים ממוצע של price בטבלת  transactions יש בעיה כי המטבע שונה מלקוח ללקוח. אם לא נתחשב במטבע, החישוב יכלול גם דולרים וגם יורו למשל, והתוצאה שתתקבל תהיה שגויה.

2)

select c.currency, avg(t.price) as avg\_price

from transactions as t

join customers as c

on t.customerID= c.customerID

group by c.currency

;

3)

כדי לחשב את ערך הקניות בדולרים, יש להוסיף טבלה חדשה בשם exchange\_rates שתכיל את סוג המטבע (currency), תאריך (date), ושער ההמרה לדולר (rate\_to\_usd). לאחר מכן, בכל שאילתה שרוצים לחשב בה את המחיר בדולרים, מצרפים את טבלת customers (כדי לדעת את המטבע של כל לקוח) ואת טבלת exchange\_rates לפי המטבע והתאריך של העסקה, ומכפילים את המחיר (price) בשער ההמרה (rate\_to\_usd) כדי לקבל את המחיר בדולרים. חשוב לציין, שצריך לבחור מראש את המטבע אותו נרצה להמיר. בדוגמא כאן בחרנו בשער הדולר $.

4)

כדי למנוע מצב שבו נרשמת עסקה (transaction) עם מוצר שאינו קיים בטבלת המוצרים (products), יש להגדיר מפתח זר (foreign key) בעמודה productID של טבלת transactions כך שיצביע על העמודה productID בטבלת products. הגדרה זו לא רק מונעת הזנה של מזהי מוצר לא תקינים, אלא גם מבטיחה שלמות ייחוסים בכך שהיא חוסמת מחיקה או עדכון של מוצר כל עוד קיימות עבורו עסקאות, ובכך מבטיחה עקביות מלאה בין שתי הטבלאות.

5)

טבלת yearmonth מאפשרת שליפות מהירות של צריכה חודשית ומועילה לניתוחים עסקיים. עם זאת, השדה Date אינו מוגדר כתאריך, אין קשר מוגדר ללקוחות, והטבלה אינה מתעדכנת אוטומטית — מה שעלול לפגוע בדיוק ובתקינות הנתונים.

6)

לשלושת השדות בטבלת transactions כדאי להוסיף constraints שמבטיחים תקינות:

* **date** – להגדיר not null כדי לוודא שאין עסקה בלי תאריך.                          בנוסף רצוי להוסיף (check (date <= current\_date כדי למנוע הכנסת עסקאות עם תאריך עתידי.
* **amount** – להוסיף  not null ו- (check (amount > 0 כדי למנוע רישום עסקאות בכמות אפסית או שלילית.
* **price** – להוסיף not null ו־(check (price >= 0 כדי למנוע מחיר שלילי (ובחלק מהמקרים אף לחייב > 0 אם אין עסקאות חינם).

הוספת constraints כאלה שומרת על דיוק ועקביות בנתוני העסקאות, מונעת טעויות הזנה בסיסיות, ותואמת את עקרונות תקינות הנתונים שהודגשו במסמך.

7)

* **בעיה א – חוסר אחידות פנימית:** השדה segment מוזן כטקסט חופשי ולכן יכולים להופיע ערכים שונים לאותו סגמנט (למשל "Private", "PRIVATE" או שגיאות כתיב). חוסר אחידות כזה מקשה על ניתוחים וגורם לפיצול מלאכותי בנתונים.
* **בעיה ב – אי־תאימות בין טבלאות:** השדה מופיע גם ב־customers וגם ב־gasstations ללא טבלת קודים מרכזית, ולכן אין ערובה לכך שהערכים בשתי הטבלאות תואמים. כך ייתכן שלקוח יסומן כ־"Business" בעוד תחנה תסומן כ־"Commercial", מה שמוביל להצלבות ושאילתות מטעות.

**פתרון:** הקמת טבלת קודים מרכזית לסגמנטים והגדרת foreign key משתי הטבלאות אליה, כדי להבטיח אחידות ושלמות ייחוסים.

8)

select

p.productid,

p.description,

c.currency,

sum(t.price \* t.amount) / nullif(sum(t.amount), 0) as avg\_unit\_price

from transactions t

join products p

on t.productid = p.productid

join customers c

on t.customerid = c.customerid

where t.amount > 0

and c.currency is not null

group by p.productid, p.description, c.currency;

9)

הפתרון הנכון הוא לשמור את מחיר המוצר בטבלת מחירים היסטורית ייעודית, למשל product\_prices, שבה לכל מוצר מוגדר מחיר יחיד בכל רגע נתון בצירוף המטבע ותוקף זמן (valid\_from, valid\_to). כך ניתן לשחזר בדיוק את המחיר הנכון ביום מסוים ולבצע ניתוחים עסקיים עקביים. שמירת המחיר ב־products אינה מאפשרת היסטוריה, ואילו ב־transactions נשמר רק המחיר בפועל (כולל הנחות וחריגות). מודל זה מבטיח עקביות, שלמות ייחוסים ותמיכה במטבעות ותאריכים, ולכן הוא הפתרון האידיאלי.

נניח מוצר דלק 95 (productid = 101). בטבלת product\_prices נשמור כך:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **productid** | **currency** | **unit\_price** | **valid\_from** | **valid\_to** |
| 101 | ILS | 6.50 | 2025-01-01 | 2025-03-31 |
| 101 | ILS | 6.70 | 2025-04-01 | 2025-06-30 |
| 101 | ILS | 6.80 | 2025-07-01 | null |

עם זאת, פתרון פשוט יותר הוא להוסיף עמודת מחיר יחידה (למשל בדולרים) בטבלת products. פתרון זה מתאים רק כאשר מניחים מחיר אחיד וקבוע למוצר, אך הוא אינו תומך בהיסטוריה של מחירים או במטבעות שונים.

10)

במבנה הנתונים הקיים בטבלת transactions, כל שורה מתארת פריט יחיד בעסקה ולכן עסקה שמכילה כמה מוצרים חייבת להתפצל לכמה שורות– אחת לכל מוצר. המשמעות היא שאין ישות ברורה של "עסקה" אחת עם כמה פריטים, אלא רצף שורות נפרדות שניתן רק לנחש אם הן קשורות לאותה קנייה על סמך שדות כמו תאריך, שעה ולקוח.

**יתרונות:** המבנה פשוט, קל לשאילתות לפי מוצר, ואין צורך בטבלת פריטים נוספת.  
**חסרונות:** אין מזהה עסקה מאחד, מה שמקשה לנתח עסקאות שלמות, עלול לגרום לכפילויות מידע (אותם נתונים חוזרים בכמה שורות), ומגביל יכולת להוסיף מידע ברמת עסקה (למשל הנחה כוללת או אמצעי תשלום).

**פתרון עדיף:** להוסיף מזהה עסקה (transactionid) – כפי שקיים בטבלת transactions\_1k – ולשמור את פרטי העסקה בטבלת־אב נפרדת, ואת רשימת המוצרים בטבלת־בן (transaction\_items). כך ניתן לייצג עסקה מלאה עם כמה מוצרים, לשמור על שלמות ייחוסים, ולמנוע כפילויות.