|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מס' קבוצה | אתר | | | תאריך הגשה |
| 15 | www.paireyewear.com | | | 19/06/2025 |
| חברי הצוות – מספרי ת.ז | | | | |
| 322691783 | | 314939604 | 211395645 | |

## פרויקט בסיסי נתונים – חלק ג'

## פרק ראשון – מטלות חובה

## 

## מטלת חובה מקדימה – תיקון ה-ERD והרחבת היקף בסיס נתונים (קנס של עד 20% לציון על ביצוע לא ראוי)

**A diagram of a computer

AI-generated content may be incorrect.מודל ERD שהוגש בחלק ב'**

## מודל טבלאי שהוגש בחלק ב'

* **CUSTOMERS** ( Customer\_ID, First\_Name, Last\_Name, Email )
* **CREDIT\_CARDS** ( Card\_Number, Expiration\_Date, CCV, Name\_On\_Card, Customer\_ID (CUSTOMERS) )
* **SEARCHES** ( DT, IP\_Address, Text, Customer\_ID (CUSTOMERS) )
* **ORDERS** ( Order ID, Date, Country, City, Street\_Name, Street\_Number, Apartment, ZIP\_Code, Prescription, Card\_Number (CREDIT\_CARDS) )
* **PRODUCTS** ( Product\_ID, Name, Description, Rating, Price )
* **ORDERABLE\_PRODUCTS** ( Product\_ID (PRODUCTS) )
* **TOP\_FRAMES** ( Product\_ID (ORDERABLE\_PRODUCTS), Color, Design )
* **ORDERED** ( Product\_ID (ORDERABLE\_PRODUCTS), Order\_ID (ORDERS), Units )
* **BASE\_FRAMES** ( Product\_ID (PRODUCTS), Size, Shape, Gender )
* **VERSIONS** ( Product\_ID (BASE\_FRAMES), Version, Description, Extra\_Price )
* **CUSTOMIZATIONS** ( Feature, Selection )
* **GLASSES\_ORDERED** ( {Product\_ID, Version} (VERSIONS), Order\_ID (ORDERS),

Units )

* **SELECTED** ( {Product\_ID, Version} (VERSIONS), {Feature, Selection}

(CUSTOMIZATIONS) )

* **REVIEWS** ( Review\_ID, Date, Headline, Text, Rating, Customer\_ID (CUSTOMERS), Product\_ID (PRODUCTS) )

## 

## הערות שניתנו בחלק ב'

* rating במוצר הוא שדה מחושב והוא הממוצע של כל הדירוגים של אותו מוצר.
* התוספת מחיר צריכה להיות בcustomization כי עבור כל התאמה יש מחיר אחר. איך תדעו את התוספת עבור כולם?
* צריך להיות קשר בין חיפוש למוצר שהמוצר הוא מה שחזר בחיפוש.

## מודל ERD מתוקן

## 

## מודל טבלאי מתוקן

* **CUSTOMERS** ( Customer\_ID, First\_Name, Last\_Name, Email )
* **CREDIT\_CARDS** ( Card\_Number, Expiration\_Date, CCV, Name\_On\_Card, Customer\_ID (CUSTOMERS) )
* **SEARCHES** ( DT, IP\_Address, Text, Customer\_ID (CUSTOMERS) )
* **ORDERS** ( Order ID, Date, Country, City, Street\_Name, Street\_Number, Apartment, ZIP\_Code, Prescription, Card\_Number (CREDIT\_CARDS) )
* **PRODUCTS** ( Product\_ID, Name, Description, Price )
* **SEARCHES\_FOR\_PRODUCTS** ( Product\_ID (PRODUCTS),

{DT, IP\_Address} (SEARCHES) )

* **ORDERABLE\_PRODUCTS** ( Product\_ID (PRODUCTS) )
* **TOP\_FRAMES** ( Product\_ID (ORDERABLE\_PRODUCTS), Color, Design )
* **ORDERED** ( Product\_ID (ORDERABLE\_PRODUCTS), Order\_ID (ORDERS), Units )
* **BASE\_FRAMES** ( Product\_ID (PRODUCTS), Size, Shape, Gender )
* **VERSIONS** ( Product\_ID (BASE\_FRAMES), Version, Description )
* **CUSTOMIZATIONS** ( Feature, Selection, Extra\_Price )
* **GLASSES\_ORDERED** ( {Product\_ID, Version} (VERSIONS), Order\_ID (ORDERS),

Units )

* **SELECTED** ( {Product\_ID, Version} (VERSIONS), {Feature, Selection}

(CUSTOMIZATIONS) )

* **REVIEWS** ( Review\_ID, Date, Headline, Text, Rating, Customer\_ID (CUSTOMERS), Product\_ID (PRODUCTS) )

## מטלה 1 (35%) – שאילתות

# שתי שאילתות SELECT ללא קינון (5%, 2.5% לכל שאילתה)

**שאילתה מס' 1:**

**מי הם הלקוחות שהזמינו משקפיים בלבד (לא כולל מוצרים נלווים) ב-3 שנים האחרונות בעלות כוללת של מעל 2000 דולר?**

שאילתה זו יכולה לסייע לעסק בזיהוי לקוחות רווחיים במיוחד אשר רוכשים משקפיים בסכומים גבוהים ובכך לסייע בהצעת הצטרפות למועדון VIP, קשר אישי למניעת נטישה למתחרים וכדומה.

השאילתה:

SELECT DISTINCT C.Customer\_ID, SUM(P.Price\*G\_O.Units +ISNULL(CS.Extra\_Price, 0)) AS TC

FROM ORDERS O JOIN CREDIT\_CARDS CC ON O.Card\_Number = CC.Card\_Number

JOIN CUSTOMERS C ON C.Customer\_ID=CC.Customer\_ID

JOIN GLASSES\_ORDERED G\_O ON G\_O.Order\_ID=O.Order\_ID

JOIN PRODUCTS P ON G\_O.Product\_ID=P.Product\_ID

JOIN VERSIONS V ON V.Product\_ID=P.Product\_ID AND G\_O.[Version] =V.[Version]

JOIN SELECTED S ON S.[Version] =V.[Version] AND S.Product\_ID =V.Product\_ID

JOIN CUSTOMIZATIONS CS ON CS.Feature=S.Feature AND S.Selection = CS.Selection

WHERE DATEDIFF(YEAR, O.Date,GETDATE()) <=3

GROUP BY C.Customer\_ID, C.First\_Name, c.Last\_Name

HAVING SUM(G\_O.Units\*P.Price)>2000

ORDER BY TC

פלט השאילתה:



**שאילתה מס' 2:**

**מי הם 10 המוצרים הכי נרכשים באתר מתחילת שנת 2025?**

שאילתה זו יכולה לעזור לעסק לאבחן מוצרים פופולריים שלקוחות נטו לרכוש בשנה האחרונה ולהציע אותם ללקוחות שטרם רכשו אותם.

השאילתה:

SELECT TOP 10 P.Name, Amount = SUM(ISNULL(O.Units, 0) + ISNULL(G.Units, 0))

FROM Products AS P

LEFT JOIN Ordered AS O ON P.Product\_ID = O.Product\_ID

LEFT JOIN Orders AS Ord1 ON O.Order\_ID = Ord1.Order\_ID

LEFT JOIN Glasses\_Ordered AS G ON P.Product\_ID = G.Product\_ID

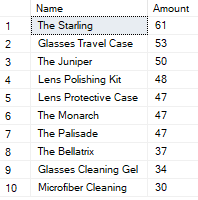
LEFT JOIN Orders AS Ord2 ON G.Order\_ID = Ord2.Order\_ID

WHERE (YEAR(Ord1.[Date]) = 2025 OR YEAR(Ord2.[Date]) = 2025)

GROUP BY P.Name

ORDER BY Amount DESC

פלט השאילתה:



# שתי שאילתות SELECT מקוננות (10%, 5% לכל שאילתה)

**שאילתה מס' 1:**

**מבין כל המוצרים שבוצע עבורם חיפוש ע"י לקוחות, איזה מוצרים לא נרכשו לעולם?**

שאילתה זו יכולה לעזור לעסק לאבחן מוצרים שלקוחות מעוניינים בהם ומחפשים אותם באתר, אך כאשר הלקוחות נכנסים לדף של מוצרים אלו - הם מתחרטים ובוחרים לא לרכוש אותם.

השאילתה:

SELECT DISTINCT P.Product\_ID, P.Name

FROM PRODUCTS AS P JOIN SEARCHES\_FOR\_PRODUCTS AS SFP ON P.Product\_ID =

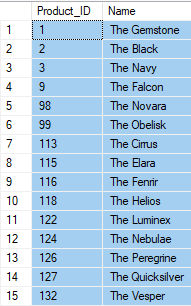
SFP.Product\_ID

WHERE P.Product\_ID NOT IN (

SELECT Product\_ID FROM ORDERED

UNION

SELECT Product\_ID FROM GLASSES\_ORDERED )

פלט השאילתה:

**שאילתה מס' 2:**

**מי הם הלקוחות שדירגו מוצר כלשהו מתחת לדירוג הממוצע של המוצר?**

שאילתה זו עשויה לסייע בזיהוי הלקוחות שדירגו מוצר באופן נמוך יחסית, באמצעות שאילתה זו ניתן לנתח את הביקורת שלהם ולבדוק האם ניתן לשפר את המוצר או חווית הרכישה

השאילתה:

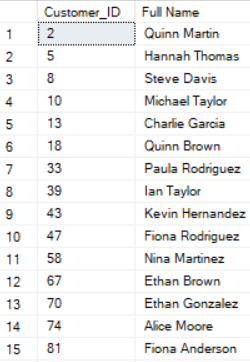
SELECT DISTINCT C.Customer\_ID, [Full Name]= c.First\_Name+' '+C.Last\_Name

FROM CUSTOMERS C JOIN REVIEWS R ON R.Customer\_ID=C.Customer\_ID

WHERE R.Rating < ( SELECT AVG(R1.Rating)

FROM REVIEWS R1

WHERE R1.Product\_ID = R.Product\_ID )

פלט השאילתה (מוצגים 15 לקוחות מתוך 54 הלקוחות שקיבלנו בפלט): 

## שאילתות עסקיות המשלבות Window Functions (10%, 2.5% לכל שימוש נכון בפונקציה)

**שאילתה מס' 1:**

**מהם הרבעונים הרווחיים ביותר בכל שנה? בנוסף, כיצד השתנה הרווח של כל רבעון בהשוואה למקבילו בשנה הקודמת?**

המטרה היא לנתח את ביצועי הרווח בשנים האחרונות, ומשאילתא זו נוכל לנתח מידע על רבעונים יותר רווחיים, ולהשליך על אסטרטגיות השיווק והחשיפה של העסק בכל אחת מהתקופות.

בנוסף, נוכל לבחון האם אנחנו במגמת עליית או ירידה, ולתת גם לנתון זה משקל בניתוח אסטרטגיות קודמות ותכנון לכאלה חדשות.

השאילתה:

SELECT

Year,

Quarter,

Quarterly\_Revenue = SUM(Revenue),

Quarter\_Rank = RANK() OVER (PARTITION BY Year ORDER BY SUM(Revenue)

DESC),

Same\_Quarter\_Last\_Year =

LAG(SUM(Revenue)) OVER (PARTITION BY Quarter ORDER BY Year),

Revenue\_Change =

SUM(Revenue) - LAG(SUM(Revenue)) OVER (PARTITION BY Quarter ORDER BY

Year)

FROM (

-- Regular product orders

SELECT

YEAR(ord.Date) AS Year,

DATEPART(QUARTER, ord.Date) AS Quarter,

Revenue = o.Units \* p.Price

FROM ORDERS ord

JOIN ORDERED o ON ord.Order\_ID = o.Order\_ID

JOIN PRODUCTS p ON o.Product\_ID = p.Product\_ID

UNION ALL

-- Customized glasses orders

SELECT

YEAR(ord.Date) AS Year,

DATEPART(QUARTER, ord.Date) AS Quarter,

Revenue = g.Units \* vp.Total\_Price

FROM ORDERS ord

JOIN GLASSES\_ORDERED g ON ord.Order\_ID = g.Order\_ID

JOIN (

-- A table of each glasses's version and it's price

SELECT

v.Product\_ID,

v.Version,

Total\_Price = p.Price + ISNULL(SUM(c.Extra\_Price), 0)

FROM VERSIONS v

JOIN PRODUCTS p ON v.Product\_ID = p.Product\_ID

LEFT JOIN SELECTED s ON v.Product\_ID = s.Product\_ID AND v.Version =

s.Version

LEFT JOIN CUSTOMIZATIONS c ON s.Feature = c.Feature AND s.Selection =

c.Selection

GROUP BY v.Product\_ID, v.Version, p.Price

) AS vp ON g.Product\_ID = vp.Product\_ID AND g.Version = vp.Version

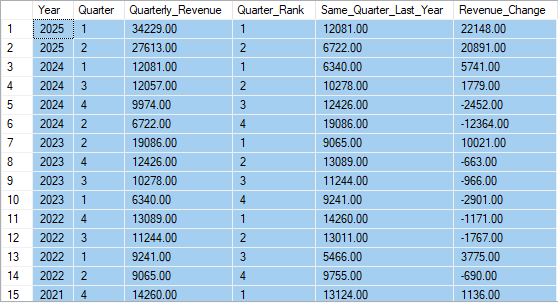
) AS All\_Revenue

WHERE Year >= YEAR(GETDATE()) - 5

GROUP BY Year, Quarter

ORDER BY Year DESC, Quarter\_Rank

פלט השאילתה (15 רשומות מתוך 22 שהתקבלו):



**שאילתה מס' 2:**

**מי הם הלקוחות שביצעו יחס גבוה של הזמנות לעומת חיפושים (מחולק לרבעים, כאשר הרביע העליון יהיה 25% הלקוחות עם יחס הזמנה-חיפוש הגבוה ביותר) ומהי תדירות החיפוש הממוצעת שלהם?**

מטרת השאילתה היא הצגת הלקוחות המראים עניין במוצרי האתר אך אך אינם מבצעים הרבה פעולות רכישה (ביחס לכמה שחיפשו), לכן יהיה נכון להניע אותם לרכישה בדרכים מגוונות (מבצעים, תמריצים וכדומה). בנוסף השאילתה מציינת את תדירות החיפוש הממוצעת, תדירות זו יכולה ללמד על הרגלי החיפוש של הלקוח ולסייע בהתאמת אסטרטגיות שיווקיות מותאמת זמן.

השאילתה:

SELECT \*

FROM (

SELECT C.Customer\_ID, C.First\_Name + ' ' + C.Last\_Name AS Full\_Name,

COUNT(DISTINCT S.DT) AS Number\_Of\_Searches,

COUNT(DISTINCT O.Order\_ID) AS Number\_Of\_Orders,

CAST(ROUND(1.0 \* COUNT(DISTINCT S.DT) / NULLIF(COUNT(DISTINCT

O.Order\_ID), 0), 3) AS DECIMAL(10,3)) AS Ratio, -- חישוב היחס בין מספר החיפושים למספר ההזמנות

NTILE(4) OVER ( ORDER BY 1.0 \* COUNT(DISTINCT S.DT) /

NULLIF(COUNT(DISTINCT O.Order\_ID), 0) DESC ) AS Quartile , -- חלוקת הלקוחות ל 4 רביעים בהתאם ליחס חיפוש-רכישה שלהם

( SELECT AVG(DATEDIFF(DAY, Prev\_Search\_Date, Search\_Date)) -- חישוב ממוצע בין חיפושי הלקוח

FROM (

SELECT S1.DT AS Search\_Date,

LAG(S1.DT) OVER (PARTITION BY S1.Customer\_ID ORDER BY S1.DT) AS

Prev\_Search\_Date -- תאריך הזמנה קודם של לקוח

FROM SEARCHES S1

WHERE S1.Customer\_ID = C.Customer\_ID

) AS Gaps

WHERE Prev\_Search\_Date IS NOT NULL

) AS Avg\_Days\_Between\_Searches

FROM CUSTOMERS C

LEFT JOIN SEARCHES S ON C.Customer\_ID = S.Customer\_ID

LEFT JOIN CREDIT\_CARDS CC ON C.Customer\_ID = CC.Customer\_ID

LEFT JOIN ORDERS O ON O.Card\_Number = CC.Card\_Number

GROUP BY C.Customer\_ID, C.First\_Name, C.Last\_Name

) AS T

WHERE Quartile = 1

ORDER BY Ratio DESC

פלט השאילתה (15 רשומות מתוך 63 סה"כ):

## שאילתה מקוננת תוך שימוש ב-CTE (פסקת WITH) (10%)

**מהן המדינות הרווחיות ביותר עבור החברה? (בהסתכלות על המדינות עם: תדירות ההזמנות הגבוהה ביותר, כמות הלקוחות הגדולה ביותר, ההכנסות הגבוהות ביותר ועוד).**

שאילתה זו תסייע לבחון את אסטרטגיית השיווק הנוכחית של העסק וכן עשויה לעזור בפיתוח אסטרטגיה קדימה - היכן להשקיע יותר ואיפה פחות.

השאילתה:

WITH

ORDER\_INTERVALS AS (

SELECT

Country,

Order\_Date = O.[Date],

Prev\_Order\_Date = LAG(O.[Date]) OVER (PARTITION BY Country ORDER BY

O.[Date]),

Days\_Since\_Last\_Order = DATEDIFF(DAY,

LAG(O.[Date]) OVER (PARTITION BY Country ORDER BY O.[Date]),

O.[Date])

FROM ORDERS O

),

AVG\_INTERVALS AS (

SELECT

Country,

Avg\_Days\_Between\_Orders = AVG(Days\_Since\_Last\_Order \* 1.0)

FROM ORDER\_INTERVALS

WHERE Days\_Since\_Last\_Order IS NOT NULL

GROUP BY Country

),

CUSTOMER\_COUNTS AS (

SELECT

O.Country,

Active\_Customers = COUNT(DISTINCT CC.Customer\_ID),

Total\_Orders = COUNT(DISTINCT O.Order\_ID)

FROM ORDERS O

JOIN CREDIT\_CARDS CC ON O.Card\_Number = CC.Card\_Number

GROUP BY O.Country

),

Version\_Prices AS (

SELECT

V.Product\_ID,

V.Version,

Total\_Price = P.Price + ISNULL((

SELECT SUM(C.Extra\_Price)

FROM SELECTED S

JOIN CUSTOMIZATIONS C ON S.Feature = C.Feature AND

S.Selection = C.Selection

WHERE S.Product\_ID = V.Product\_ID AND S.Version = V.Version

), 0)

FROM VERSIONS V

JOIN PRODUCTS P ON V.Product\_ID = P.Product\_ID

),

REVENUE\_BY\_COUNTRY AS (

SELECT

O.Country,

Total\_Revenue =

ISNULL(SUM(OD.D\_Revenue), 0) + ISNULL(SUM(OG.G\_Revenue), 0)

FROM ORDERS O

LEFT JOIN (

SELECT

D.Order\_ID,

SUM(D.Units \* P.Price) AS D\_Revenue

FROM ORDERED D

JOIN PRODUCTS P ON D.Product\_ID = P.Product\_ID

GROUP BY D.Order\_ID

) AS OD ON O.Order\_ID = OD.Order\_ID

LEFT JOIN (

SELECT

G.Order\_ID,

SUM(G.Units \* VP.Total\_Price) AS G\_Revenue

FROM GLASSES\_ORDERED G

JOIN Version\_Prices VP ON G.Product\_ID = VP.Product\_ID AND G.Version =

VP.Version

GROUP BY G.Order\_ID

) AS OG ON O.Order\_ID = OG.Order\_ID

GROUP BY O.Country

),

COMBINED\_STATS AS (

SELECT

A.Country,

Avg\_Days\_Between\_Orders,

C.Active\_Customers,

C.Total\_Orders,

R.Total\_Revenue,

Revenue\_Share = R.Total\_Revenue \* 1.0 / SUM(R.Total\_Revenue) OVER(),

Efficiency = R.Total\_Revenue \* 1.0 / NULLIF(A.Avg\_Days\_Between\_Orders, 0)

FROM AVG\_INTERVALS A

JOIN CUSTOMER\_COUNTS C ON A.Country = C.Country

JOIN REVENUE\_BY\_COUNTRY R ON A.Country = R.Country

)

SELECT

Country,

CAST(Avg\_Days\_Between\_Orders AS DECIMAL(10,2)) AS

Avg\_Days\_Between\_Orders,

Active\_Customers,

Total\_Orders,

CAST(Total\_Revenue AS DECIMAL(12,2)) AS Total\_Revenue,

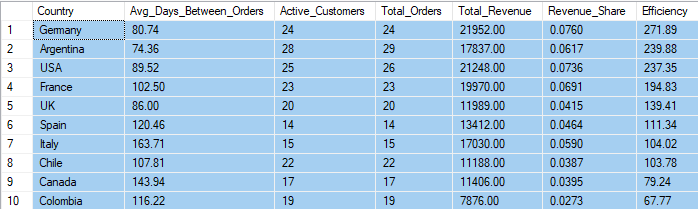
CAST(Revenue\_Share AS DECIMAL(5,4)) AS Revenue\_Share,

CAST(Efficiency AS DECIMAL(12,2)) AS Efficiency

FROM COMBINED\_STATS

ORDER BY Efficiency DESC

הפלט, מוצגות 10 רשומות מתוך 59 שחזרו סה"כ:



## מטלה 2 (35%) – יישומי כלים מתקדמים

# View (5%)

**רשימת כל המוצרים, כאשר עבור כל מוצר מוצג: דירוג, מס' החיפושים שבוצעו עבורו, מס' ההזמנות בהן הזמינו אותו, יחס חיפוש-הזמנה, מהי התדירות הממוצעת בין כל 2 הזמנות, מס' יחידות כולל שהוזמנו ממנו, מהי המדינה שהזמינה ממנו הכי הרבה.**

שאילתה זו תסייע בהצגת פידבק של רוחבי וכללי עבור כל המוצרים באתר. לכן, טבעי שהעסק יעשה שימוש בשאילתה זאת בתדירות גבוהה גם כשאילתה עצמאית וגם כתת-שאילתה עבור שאילתות אחרות.

יצירת ה-VIEW:

CREATE VIEW PRODUCTS\_OVERVIEW AS

SELECT

P.Product\_ID,

P.Name,

-- Rating

Rating = ISNULL(ROUND(AVG(CAST(R.Rating AS FLOAT)), 2), 0),

-- Search count & order count

Total\_Searches = ISNULL(SC.Total\_Searches, 0),

Total\_Orders = ISNULL(OC.Total\_Orders, 0),

-- Search to order ratio

Search\_Order\_Ratio =

ROUND (CASE

WHEN ISNULL(OC.Total\_Orders, 0) = 0 THEN 0

WHEN ISNULL(SC.Total\_Searches, 0) = 0 THEN 0

ELSE CAST(OC.Total\_Orders AS FLOAT) /

SC.Total\_Searches

END,

3),

-- Avg days between orders

Avg\_Days\_Between\_Orders =

CAST((

SELECT AVG(DATEDIFF(DAY, PrevOrder, CurrOrder) \* 1.0)

FROM ( SELECT

O.Date AS CurrOrder,

LAG(O.Date) OVER (ORDER BY O.Date ASC) AS

PrevOrder

FROM ORDERS O

WHERE O.Order\_ID IN (

SELECT Order\_ID FROM ORDERED WHERE

Product\_ID = P.Product\_ID

UNION

SELECT Order\_ID FROM GLASSES\_ORDERED

WHERE Product\_ID = P.Product\_ID

)

) AS Gaps

WHERE PrevOrder IS NOT NULL

) AS DECIMAL(10,2)),

-- Total units

Total\_Units = ISNULL((

SELECT SUM(Units)

FROM (

SELECT Units FROM ORDERED WHERE Product\_ID = P.Product\_ID

UNION ALL

SELECT Units FROM GLASSES\_ORDERED WHERE Product\_ID = P.Product\_ID

) AS AllUnits

), 0),

-- Top country

Top\_Country =

(

SELECT TOP 1 O.Country

FROM (

SELECT Order\_ID

FROM ORDERED

WHERE Product\_ID = P.Product\_ID

UNION ALL

SELECT Order\_ID

FROM GLASSES\_ORDERED

WHERE Product\_ID = P.Product\_ID

) AS OrdersList

JOIN ORDERS O ON OrdersList.Order\_ID = O.Order\_ID

GROUP BY O.Country

ORDER BY COUNT(\*) DESC)

FROM PRODUCTS P

-- Ratings

LEFT JOIN REVIEWS R ON P.Product\_ID = R.Product\_ID

-- Precomputed search count per product

LEFT JOIN (

SELECT Product\_ID, Total\_Searches = COUNT(\*)

FROM SEARCHES\_FOR\_PRODUCTS

GROUP BY Product\_ID

) SC ON SC.Product\_ID = P.Product\_ID

-- Precomputed orders count per product

LEFT JOIN (

SELECT Product\_ID, Total\_Orders = COUNT(\*)

FROM (

SELECT Product\_ID, Order\_ID FROM ORDERED

UNION ALL

SELECT Product\_ID, Order\_ID FROM GLASSES\_ORDERED

) AS AllOrders

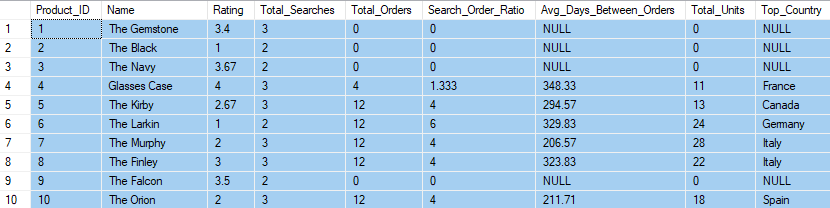
GROUP BY Product\_ID

) OC ON OC.Product\_ID = P.Product\_ID

GROUP BY P.Product\_ID, P.Name, SC.Total\_Searches, OC.Total\_Orders

פלט כללי של ה-VIEW (מוצגים 10 רשומות מתוך 134):

SELECT \*

FROM PRODUCTS\_OVERVIEW

דוגמא לשימוש פרקטי של ה-VIEW:

נרצה לענות על השאילתה: מי הם עשרת המוצרים עם הדירוג הגבוה ביותר באתר, כמה הזמנות בוצעו עבור כל אחד ממוצרים אלה ומה היא המדינה ממנה כל מוצר הוזמן הכי הרבה?

ללא ה-VIEW:

SELECT TOP 10

P.Product\_ID,

P.Name,

Rating = ROUND(AVG(CAST(R.Rating AS FLOAT)), 2),

Total\_Orders = (

SELECT COUNT(\*)

FROM (

SELECT Order\_ID FROM ORDERED WHERE Product\_ID = P.Product\_ID

UNION ALL

SELECT Order\_ID FROM GLASSES\_ORDERED WHERE Product\_ID =

P.Product\_ID

) AS AllOrders

),

Top\_Country =

( SELECT TOP 1 O.Country

FROM (

SELECT Order\_ID FROM ORDERED WHERE Product\_ID = P.Product\_ID

UNION ALL

SELECT Order\_ID FROM GLASSES\_ORDERED WHERE Product\_ID =

P.Product\_ID

) AS OrdersList

JOIN ORDERS O ON OrdersList.Order\_ID = O.Order\_ID

GROUP BY O.Country

ORDER BY COUNT(\*) DESC

)

FROM PRODUCTS P

JOIN REVIEWS R ON P.Product\_ID = R.Product\_ID

GROUP BY P.Product\_ID, P.Name

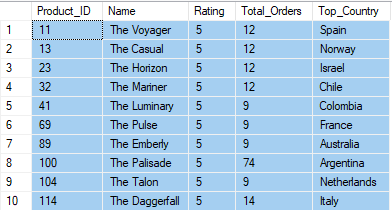
ORDER BY Rating DESC

עם ה-VIEW:

SELECT TOP 10 Product\_ID, Name, Rating, Total\_Orders, Top\_Country

FROM PRODUCTS\_OVERVIEW

ORDER BY Rating DESC

פלט השאילתה (כמובן זהה עם וללא ה-VIEW):

# פונקציות (Functions) (10%, 2 פונקציות – 5% לכל פונקציה)

**פונקציה מס' 1:**

**פונ' המקבלת מזהה ייחודי של Review ומחזירה את ההפרש בין הדירוג שקיבל המוצר ב-Review לבין הדירוג הכללי של המוצר.**

הבהרה לגבי פלט הפונקציה:

כאשר המס' המתקבל חיובי - הביקורת החזירה דירוג גבוה יותר מאשר הדירוג הממוצע. אם המס' המתקבל שלילי - הביקורת החזירה דירוג נמוך יותר מאשר הדירוג הממוצע.

הצדקה להוספת הפונקציה:

פונקציה זו יכולה לסייע באבחון ביקורות שדירגו מוצרים באופן נמוך משמעותית משאר הביקורות, או לחילופין באופן גבוה משמעותית משאר הלקוחות.

דוגמא לשימוש פרקטי של הפונקציה: כתיבת TRIGGER שמריץ את הפונקציה עבור כל ביקורת שנכנסת לאתר. במידה ופלט הפונקציה הוא מס' שלילי שקטן ממינוס 3, סימן שהלקוח שכתב את הביקורת היה ממש לא מרוצה מהמוצר בהשוואה ללקוחות אחרים. מחלקת שירות הלקוחות תקבל התראה על כך ותשלח מייל ללקוח שכתב את הביקורת.

*\*חשוב לציין - בפונקציה זו נבצע שימוש ב-VIEW שהגדרנו בסעיף הקודם.*

יצירת הפונקציה:

CREATE FUNCTION GetRatingGap(@Review\_ID INT)

RETURNS FLOAT

AS

BEGIN

DECLARE @Result FLOAT

SET @Result = ( SELECT R.Rating - PO.Rating

FROM PRODUCTS\_OVERVIEW AS PO JOIN REVIEWS AS R ON

PO.Product\_ID = R.Product\_ID

WHERE R.Review\_ID = @Review\_ID )

RETURN @Result

END

דוגמא להפעלת הפונקציה:

בשנה הנוכחית, עבור אילו ביקורות הפער בין הדירוג הניתן בביקורת לדירוג המוצר קטן ממינוס 2?

SELECT \*, Gap = dbo.GetRatingGap(Review\_ID)

FROM REVIEWS

WHERE YEAR(Date) = YEAR(GetDate()) AND dbo.GetRatingGap(Review\_ID) < -2

פלט השאילתה (משיקולי אסתטיקה, לא הצגנו את השדה Text במלואו):



**פונקציה מס' 2:**

**פונקציה המקבלת מזהה ייחודי של לקוח ומחזירה את כל היסטוריית ההזמנות שלו.**

הצדקה להוספת הפונקציה:

הפונ' רלוונטית עבור מקרים בהם נרצה להפיק מידע על היסטוריית ההזמנות של לקוח מסוים.

דוגמא לשימוש פרקטי של הפונקציה: לקוח מתקשר לשירות הלקוחות של האתר בטענה כי משלוח שהזמין לפני יותר מחודשיים לא הגיע. אחת מהפעולות הראשונות שנצפה מנציג שירות הלקוחות לבצע תהיה הפעלת הפונקציה כאשר הקלט הוא מזהה הלקוח. במידה והפונקציה לא הייתה קיימת, זה היה מצריך מנציג שירות הלקוחות לכתוב שאילתה כדי לקבל את היסטוריית ההזמנות של הלקוח וכתוצאה מכך לעכב את מתן השירות ללקוח.

יצירת הפונקציה:

CREATE FUNCTION GetCustomerHistory (@CustomerID INT)

RETURNS TABLE

AS

RETURN (

WITH Version\_Prices AS (

SELECT

V.Product\_ID,

V.Version,

P.Price + ISNULL(SUM(C.Extra\_Price), 0) AS Total\_Price

FROM VERSIONS V

JOIN BASE\_FRAMES BF ON V.Product\_ID = BF.Product\_ID

JOIN PRODUCTS P ON BF.Product\_ID = P.Product\_ID

LEFT JOIN SELECTED S ON V.Product\_ID = S.Product\_ID AND V.Version =

S.Version

LEFT JOIN CUSTOMIZATIONS C ON S.Feature = C.Feature AND S.Selection =

C.Selection

GROUP BY V.Product\_ID, V.Version, P.Price

),

Ordered\_Summary AS (

SELECT

O.Order\_ID,

SUM(ISNULL(D.Units, 0)) AS Units\_Ordered,

SUM(ISNULL(D.Units \* P.Price, 0)) AS Cost\_Ordered

FROM ORDERS O

JOIN ORDERED D ON O.Order\_ID = D.Order\_ID

JOIN PRODUCTS P ON D.Product\_ID = P.Product\_ID

GROUP BY O.Order\_ID

),

Glasses\_Summary AS (

SELECT

O.Order\_ID,

SUM(ISNULL(G.Units, 0)) AS Units\_Glasses,

SUM(ISNULL(G.Units \* VP.Total\_Price, 0)) AS Cost\_Glasses

FROM ORDERS O

JOIN GLASSES\_ORDERED G ON O.Order\_ID = G.Order\_ID

JOIN Version\_Prices VP ON G.Product\_ID = VP.Product\_ID AND G.Version =

VP.Version

GROUP BY O.Order\_ID

)

SELECT

O.Order\_ID,

O.[Date],

Total\_Units =

ISNULL(OS.Units\_Ordered, 0) + ISNULL(GS.Units\_Glasses, 0),

Total\_Cost =

ISNULL(OS.Cost\_Ordered, 0) + ISNULL(GS.Cost\_Glasses, 0)

FROM CREDIT\_CARDS CC

JOIN ORDERS O ON CC.Card\_Number = O.Card\_Number

LEFT JOIN Ordered\_Summary OS ON O.Order\_ID = OS.Order\_ID

LEFT JOIN Glasses\_Summary GS ON O.Order\_ID = GS.Order\_ID

WHERE CC.Customer\_ID = @CustomerID

)

דוגמה להפעלת הפונקציה:

מהי היסטוריית ההזמנות עבור לקוח עם המזהה 3?

SELECT \*

FROM dbo.GetCustomerHistory(3)

פלט השאילתה:

# Trigger (10%)

יצרנו טריגר המעדכן מחיר של כל הזמנה לאחר הוספת או מחיקת מוצר כלשהו מההזמנה. מכיוון שאצלנו ישנן 2 טבלאות אליהן נכנסות ההזמנות (טבלת ORDERED עבור מוצרים שאינם משקפיים וטבלת GLASSES\_ORDERED עבור מוצרי משקפיים) וניתן "להאזין" רק לטבלה אחת, ביצענו 2 טריגרים, אחד עבור כל טבלה, אשר מעדכנים את סכום ההזמנה הכולל.

ראשית נוסיף שדה Total\_Amount לטבלת ORDERS עם ערך דיפולטיבי 0:

ALTER TABLE ORDERS

ADD Total\_Amount DECIMAL(10,2)

CONSTRAINT DF\_ORDERS\_Total\_Amount DEFAULT 0 WITH VALUES;

כעת נעדכן את ערך Total\_Amount לערך האמיתי שלו עבור כל הזמנה:

UPDATE ORDERS

SET Total\_Amount =

ISNULL(RP.Total\_Regular, 0) + ISNULL(GO.Total\_Glasses, 0)

FROM ORDERS

LEFT JOIN (

SELECT

ORDERED.Order\_ID,

SUM(ORDERED.Units \* PRODUCTS.Price) AS Total\_Regular

FROM ORDERED

JOIN PRODUCTS ON ORDERED.Product\_ID = PRODUCTS.Product\_ID

GROUP BY ORDERED.Order\_ID

) RP ON ORDERS.Order\_ID = RP.Order\_ID

LEFT JOIN (

SELECT

GLASSES\_ORDERED.Order\_ID,

SUM(GLASSES\_ORDERED.Units \* (PRODUCTS.Price + ISNULL(C.Extra\_Total,

0))) AS Total\_Glasses

FROM GLASSES\_ORDERED

JOIN PRODUCTS ON GLASSES\_ORDERED.Product\_ID = PRODUCTS.Product\_ID

LEFT JOIN (

SELECT

SELECTED.Product\_ID,

SELECTED.Version,

SUM(CUSTOMIZATIONS.Extra\_Price) AS Extra\_Total

FROM SELECTED

JOIN CUSTOMIZATIONS ON SELECTED.Feature = CUSTOMIZATIONS.Feature

AND SELECTED.Selection =

CUSTOMIZATIONS.Selection

GROUP BY SELECTED.Product\_ID, SELECTED.Version

) C ON GLASSES\_ORDERED.Product\_ID = C.Product\_ID AND

GLASSES\_ORDERED.Version = C.Version

GROUP BY GLASSES\_ORDERED.Order\_ID

) GO ON ORDERS.Order\_ID = GO.Order\_ID

כעת נכתוב טריגר לחישוב שדה Total\_Amount כתוצאה מהוספה / מחיקה מ-ORDERED:

CREATE TRIGGER UpdateAmountOrderableProducts

ON ORDERED

AFTER INSERT, DELETE

AS

BEGIN

-- במידה ומתבצעות הוספת רשומות

IF EXISTS (SELECT \* FROM INSERTED)

BEGIN

UPDATE O

SET O.Total\_Amount = O.Total\_Amount + X.TotalPrice

FROM ORDERS O

JOIN (

SELECT I.Order\_ID, SUM(I.Units \* P.Price) AS TotalPrice

FROM INSERTED I

JOIN PRODUCTS P ON I.Product\_ID = P.Product\_ID

GROUP BY I.Order\_ID

) AS X ON O.Order\_ID = X.Order\_ID

END

-- במידה ומתבצעות מחיקת רשומות

IF EXISTS (SELECT \* FROM DELETED)

BEGIN

UPDATE O

SET O.Total\_Amount = O.Total\_Amount - X.TotalPrice

FROM ORDERS O

JOIN (

SELECT D.Order\_ID, SUM(D.Units \* P.Price) AS TotalPrice

FROM DELETED D

JOIN PRODUCTS P ON D.Product\_ID = P.Product\_ID

GROUP BY D.Order\_ID

) AS X ON O.Order\_ID = X.Order\_ID

END

END

ולבסוף טריגר על טבלת GLASSES ORDERED:

## 

CREATE TRIGGER UpdateAmountGlasses

ON GLASSES\_ORDERED

FOR INSERT, DELETE

AS

BEGIN

-- במידה ומתבצעות הוספת רשומות

IF EXISTS (SELECT \* FROM INSERTED)

BEGIN

WITH ExtraPerUnit AS (

SELECT

I.Order\_ID,

I.Product\_ID,

I.Version,

SUM(ISNULL(CS.Extra\_Price, 0)) AS Extra

FROM INSERTED I

JOIN VERSIONS V ON V.Product\_ID = I.Product\_ID AND V.Version =

I.Version

JOIN SELECTED S ON S.Product\_ID = V.Product\_ID AND S.Version =

V.Version

JOIN CUSTOMIZATIONS CS ON CS.Feature = S.Feature AND CS.Selection

= S.Selection

GROUP BY I.Order\_ID, I.Product\_ID, I.Version

)

UPDATE O

SET O.Total\_Amount = ISNULL(O.Total\_Amount, 0) + X.TotalPrice

FROM ORDERS O

JOIN (

SELECT

I.Order\_ID,

SUM(I.Units \* (P.Price + ISNULL(E.Extra, 0))) AS TotalPrice

FROM INSERTED I

JOIN PRODUCTS P ON I.Product\_ID = P.Product\_ID

LEFT JOIN ExtraPerUnit E ON I.Order\_ID = E.Order\_ID AND

I.Product\_ID = E.Product\_ID AND I.Version = E.Version

GROUP BY I.Order\_ID

) AS X ON O.Order\_ID = X.Order\_ID;

END

-- במידה ומתבצעות מחיקת רשומות

IF EXISTS (SELECT \* FROM DELETED)

BEGIN

WITH ExtraPerUnit AS (

SELECT

D.Order\_ID,

D.Product\_ID,

D.Version,

SUM(ISNULL(CS.Extra\_Price, 0)) AS Extra

FROM DELETED D

JOIN VERSIONS V ON V.Product\_ID = D.Product\_ID AND V.Version =

D.Version

JOIN SELECTED S ON S.Product\_ID = V.Product\_ID AND S.Version =

V.Version

JOIN CUSTOMIZATIONS CS ON CS.Feature = S.Feature AND CS.Selection

= S.Selection

GROUP BY D.Order\_ID, D.Product\_ID, D.Version

)

UPDATE O

SET O.Total\_Amount = ISNULL(O.Total\_Amount, 0) - X.TotalPrice

FROM ORDERS O

JOIN (

SELECT

D.Order\_ID,

SUM(D.Units \* (P.Price + ISNULL(E.Extra, 0))) AS TotalPrice

FROM DELETED D

JOIN PRODUCTS P ON D.Product\_ID = P.Product\_ID

LEFT JOIN ExtraPerUnit E ON D.Order\_ID = E.Order\_ID AND

D.Product\_ID = E.Product\_ID AND D.Version = E.Version

GROUP BY D.Order\_ID

) AS X ON O.Order\_ID = X.Order\_ID;

END

END

דוגמא להפעלת הטריגר:

לקוח בשם מיכאל מרטינז יוצר קשר עם שירות הלקוחות.  
הוא מסביר שהזמין בטעות זוג משקפיים בהתאמה אישית מההזמנה האחרונה שלו - אך הוא אינו יודע את מזהה ההזמנה. הוא כן יודע את כתובת האימייל שלו: [michael.martinez14@outlook.org](mailto:michael.martinez14@outlook.org). בנוסף, לאחר שתיאר מיכאל את זוג המשקפיים לשירות הלקוחות, הנציג זיהה שמדובר ב:

ProductID=14, Version=1.

כעת, נציג השירות צריך למצוא את המזהה של ההזמנה האחרונה של הלקוח לפי כתובת האימייל, ואז למחוק ממנה את פרטי המשקפיים הספציפיים.

לפני הרצת השאילתה:

השאילתה:

DELETE FROM GLASSES\_ORDERED

WHERE Product\_ID = 14

AND Version = 1

AND Order\_ID = (

SELECT TOP 1 O.Order\_ID

FROM ORDERS O

JOIN CREDIT\_CARDS CC ON O.Card\_Number = CC.Card\_Number

WHERE CC.Customer\_ID = (

SELECT Customer\_ID

FROM CUSTOMERS

WHERE Email = 'michael.martinez14@outlook.org'

)

ORDER BY O.Date DESC

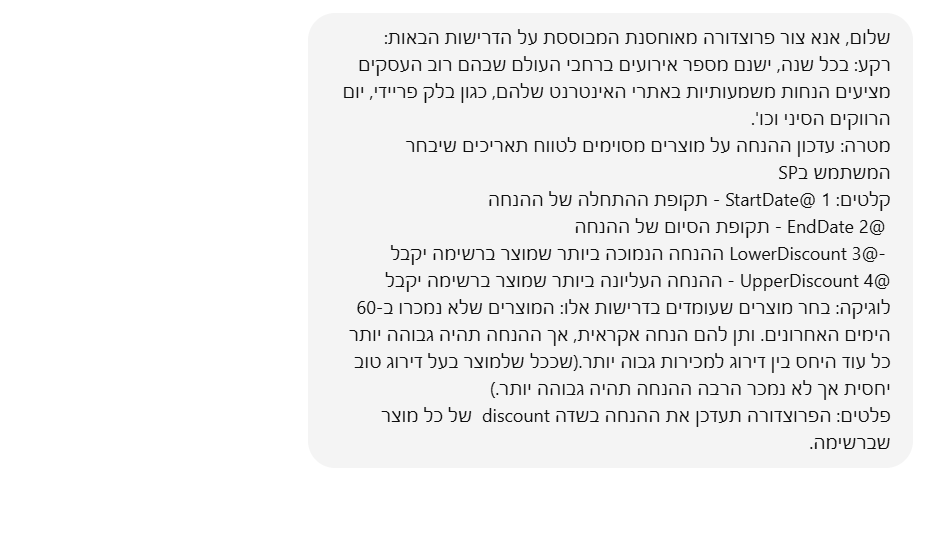
)

לאחר הרצת השאילתה:



כפי שניתן לראות, הטריגר הופעל והשדה המחושב עודכן בהתאם.

# פרוצדורה שמורה ((Stored Procedure (10%)

פרומפט שניתן ל-ChatGPT:

הקדמה לפרוצדורה - נוסיף שדה Discount:

ALTER TABLE PRODUCTS

ADD Discount Float

הפרוצדורה:

CREATE PROCEDURE ApplyDiscountsOnLowSellingProducts

@StartDate DATE,

@EndDate DATE,

@LowerDiscount FLOAT,

@UpperDiscount FLOAT

AS

BEGIN

-- עדכון זמני של כל המוצרים לפי יחס דירוג למכירות

WITH ProductStats AS (

SELECT

P.Product\_ID,

AVG(CAST(R.Rating AS FLOAT)) AS AvgRating,

ISNULL(SUM(O.Units), 0) AS TotalUnits,

DATEDIFF(DAY, MAX(Ors.Date), GETDATE()) AS DaysSinceLastOrder

FROM PRODUCTS P

LEFT JOIN REVIEWS R ON P.Product\_ID = R.Product\_ID

LEFT JOIN ORDERED O ON P.Product\_ID = O.Product\_ID

LEFT JOIN ORDERS Ors ON O.Order\_ID = Ors.Order\_ID

GROUP BY P.Product\_ID

),

RankedStats AS (

SELECT \*,

-- מחשבים את המקסימום והטווח

MAX(CASE WHEN TotalUnits = 0 THEN AvgRating ELSE AvgRating / TotalUnits END)

OVER() AS MaxRatio,

MIN(CASE WHEN TotalUnits = 0 THEN AvgRating ELSE AvgRating / TotalUnits END)

OVER() AS MinRatio

FROM ProductStats

WHERE DaysSinceLastOrder >= 150 OR DaysSinceLastOrder IS NULL

)

UPDATE P

SET Discount = ROUND(

CASE

WHEN RS.MaxRatio = RS.MinRatio THEN @UpperDiscount -- להימנע מחלוקה ב־0

ELSE @LowerDiscount +

((CASE WHEN RS.TotalUnits = 0 THEN RS.AvgRating ELSE RS.AvgRating / RS.TotalUnits END - RS.MinRatio)

/ NULLIF(RS.MaxRatio - RS.MinRatio, 0))

\* (@UpperDiscount - @LowerDiscount)

END ,2)

FROM PRODUCTS P

JOIN RankedStats RS ON P.Product\_ID = RS.Product\_ID

END

הפעלה של הפרוצדורה:

EXEC ApplyDiscountsOnLowSellingProducts

@StartDate = '2025-06-01',

@EndDate = '2025-06-30',

@LowerDiscount = 0.1,

@UpperDiscount = 0.5;

לפניי הפעלת הפרוצדורה:

מוצג 10 רשומות מתוך 134 רשומות (מוצרים)



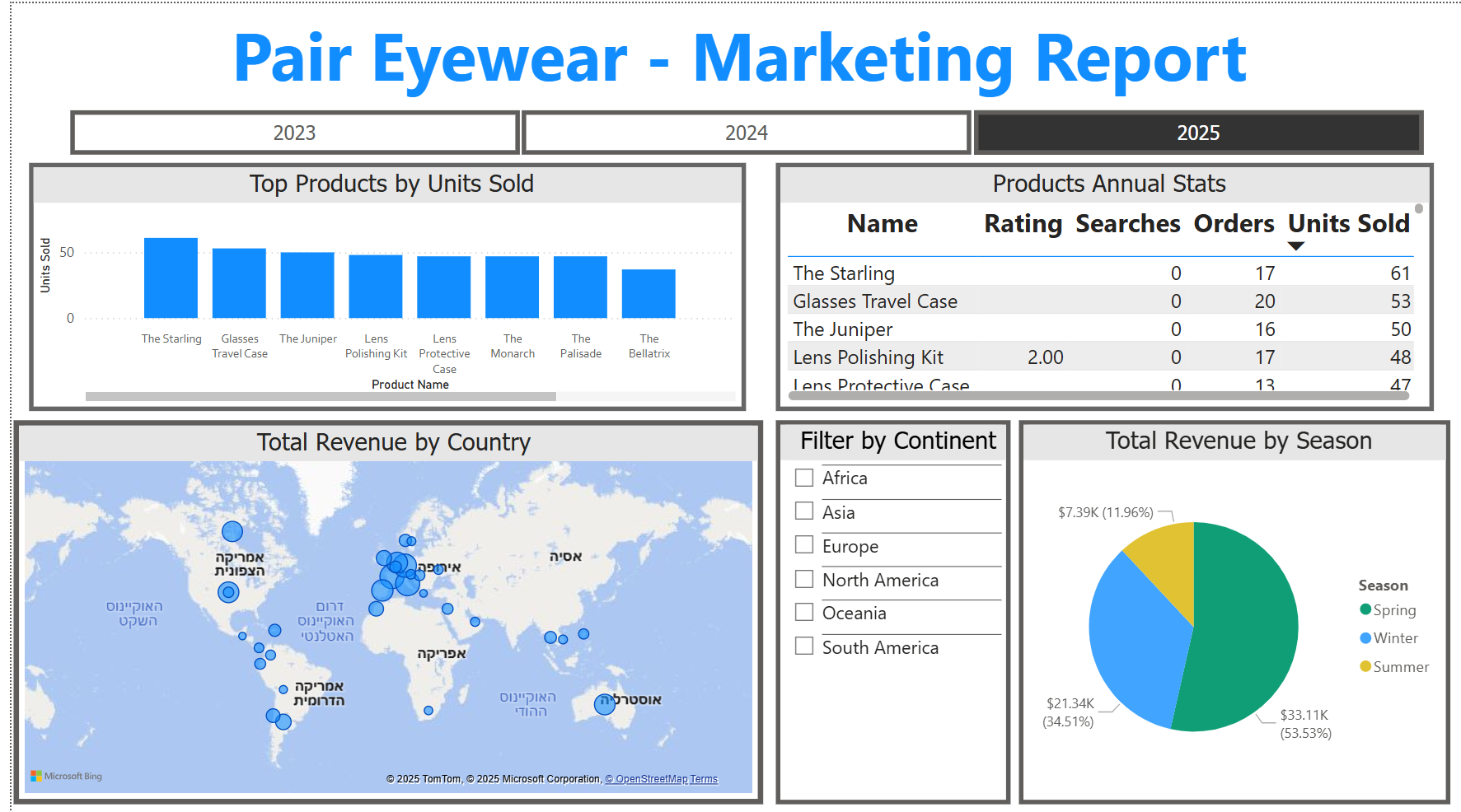
אחריי הפעלת הפרוצדורה:

מוצג 20 רשומות מתוך 134 רשומות (מוצרים)



## מטלה 3 (20%) – כלים להצגת נתונים

דו"ח עסקי למחלקת השיווק (10%)

נציג את הדו"ח העסקי עבור מחלקת השיווק של חברת Pair Eyewear:

הדו"ח נועד לנתח את דפוסי הרכישה באתר החברה, ומטרתו לספק למחלקת השיווק תובנות מבוססות נתונים לגבי ההעדפות של הלקוחות, זיהוי מגמות צרכניות, ומוצרים בעלי פוטנציאל שיווקי שטרם מומש.

הדו"ח תומך בתכנון קמפיינים שיווקיים ממוקדים ובהתאמת אסטרטגיות השיווק לצרכים המשתנים של קהל הלקוחות, במטרה להגדיל את הביקוש ולחזק את נאמנות הלקוחות.

מרכיבי הדו"ח העסקי:

סך ההזמנות לפי מדינה עם פילוח לפי יבשת - מפה אינטראקטיבית המציגה את מספר ההזמנות שבוצעו בכל מדינה, עם אפשרות סינון לפי יבשת. תובנות אלו מסייעות למחלקת השיווק לזהות אזורים גאוגרפיים בעלי ביקוש גבוה למוצרי החברה, ולהתמקד בהם בקמפיינים שיווקיים ממוקדים לצורך הגדלת נתח השוק.

פילוח עונתי של הזמנות- תרשים עוגה מציג את חלוקת ההזמנות לפי עונות השנה (Winter, Spring, Summer, Autumn), במטרה להבין מגמות עונתיות בהעדפת הצרכנים. תובנה זו תורמת להתאמת מאמצי השיווק לפי תקופות שיא ושפל לאורך השנה.

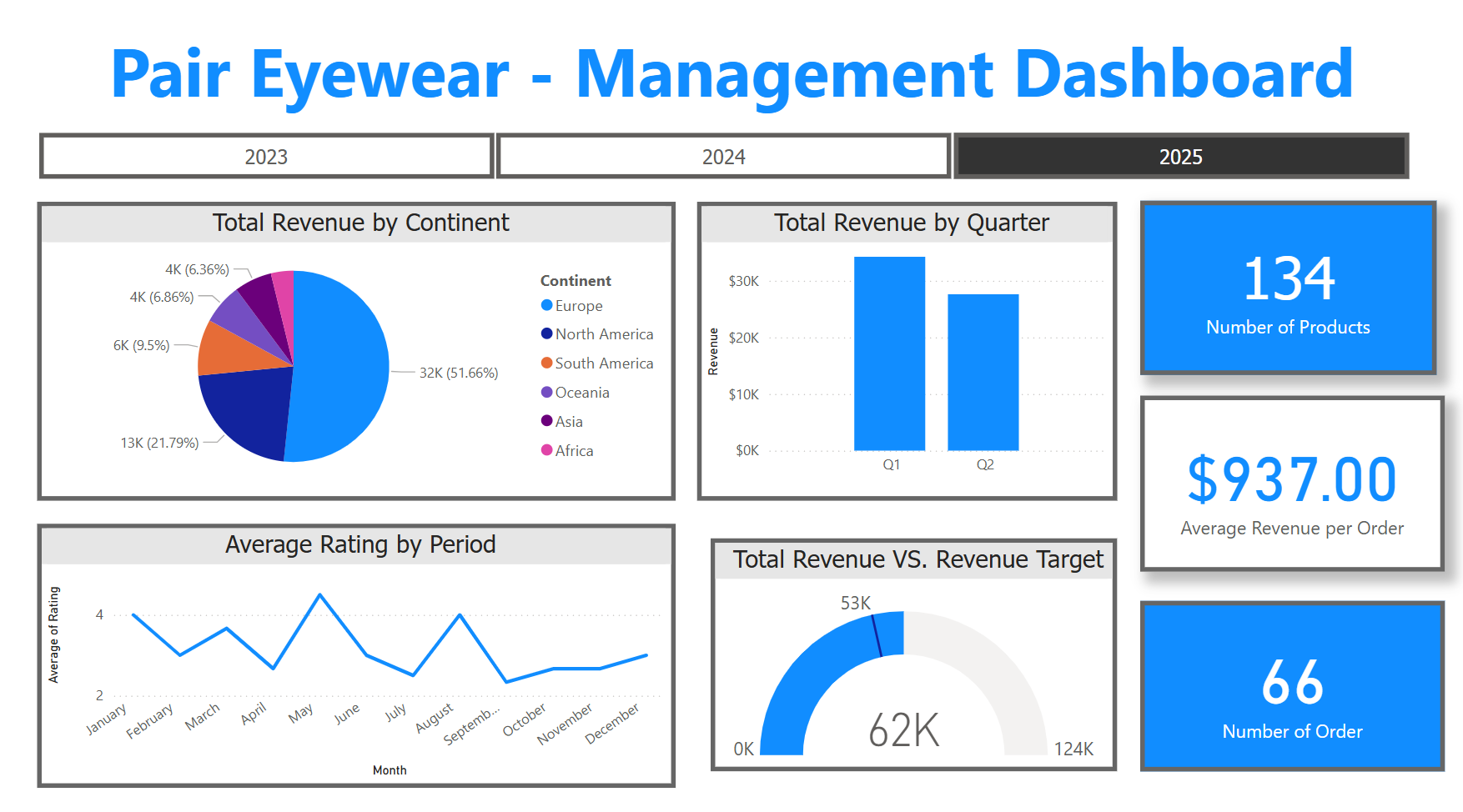
המוצרים הנמכרים ביותר- תרשים עמודות מציג את עשרת המוצרים הנמכרים ביותר. ניתוח זה מאפשר למחלקת השיווק לזהות מוצרים מובילים, לקדם אותם באופן בולט, ולגבש תובנות לגבי העדפות הלקוחות.

טבלת כלל המוצרים ונתוני הביצועים שלהם- טבלה המציגה עבור כל מוצר את: הדירוג שלו, מס' החיפושים שבוצעו עבורו, מס' ההזמנות בהן הוא הוזמן וסה"כ יחידות שנמכרו ממנו. בעזרת טבלה זו מחלקת השיווק תוכל לזהות מוצרים עם פוטנציאל שטרם מומש או לחילופין מוצר שאין טעם להשקיע בו תקציב שיווקי.

פילוח לפי שנה (2023–2025) - כלל המרכיבים בדוח ניתנים לפילוח דינמי לפי שנה באמצעות סלקטור, ובכך מאפשרים לבחון מגמות משתנות לאורך זמן ולשפר את קבלת ההחלטות השיווקית בהתבסס על נתוני עבר.

באמצעות תובנות אלו, מחלקת השיווק יכולה לבנות קמפיינים ממוקדים ומבוססי נתונים, לחדד את אסטרטגיות המכירה, ולהתאים את מגוון המוצרים וההיצע לצרכים המשתנים של הלקוחות ולמגמות השוק.

לוח מחוונים (10%)

נציג את הלוח המחוונים עבור הדרג הניהולי של חברת Pair Eyewear:

לוח המחוונים מספק סקירה מקיפה ויזואלית על הביצועים העסקיים של Pair Eyewear בשלוש השנים האחרונות (2023–2025). מטרת הלוח היא לאפשר קבלת החלטות ניהוליות מושכלות, תוך הסתמכות על נתונים עדכניים, מגמות זמן והשוואה ליעדים אסטרטגיים.

\*הבהרה - תחת "*Total Revenue By Quarter*" מופיעים רק *Q*1 ו-*Q*2 היות ובחרנו בפילוח עבור שנת 2025 וכעת אנחנו נמצאים בסוף חודש יוני.

מרכיבי לוח המחוונים:

ממוצע הדירוגים של ביקורות מוצרים באתר כפונקציה של הזמן - מדד המאפשר בחינת מגמות בשביעות רצון הלקוחות, עם אפשרות Drill Down לרמת רבעונים וחודשים. כלי זה מסייע בזיהוי מוקדם של שינויים באיכות המוצר או חוויית הלקוח.

פילוח רווח כולל לפי רבעונים **-** גרף המציג את הרווחים המצטברים ברבעונים, לצורך ניתוח עונתיות והבנת תקופות שיא בביצועים העסקיים.

פילוח רווח כולל לפי אזור גאוגרפי- תרשים עוגה המאפשר בחינה של רווח החברה לפי פילוח גאוגרפי. יש תמיכה ב-Drill Down מרמת היבשות ועד רמת הערים.

השוואת רווח בפועל מול רווח יעד **-** כלי השוואתי המציג את הרווח הכולל לצד רווח היעד (המבוסס על גידול של 30% מהשנה הקודמת). מאפשר ניטור עמידה ביעדים שנתיים ברמה ארגונית.

סה"כ מוצרים פעילים **-** אינדיקציה למגוון ההיצע שהחברה מספקת - מאפשר להבין את רוחב הקטלוג של החברה ומסייע לביסוס קשר בין היצע לביצועים.

ממוצע שווי להזמנה **-** מדד אסטרטגי לבחינת הרגלי הצריכה של הלקוחות, התומך בהחלטות בנוגע למחיר, מבצעים ומסלולי מכירה.

סה"כ הזמנות- נתון הממחיש את היקף הפעילות הכולל של החברה.

פילוח לפי 3 השנים האחרונות- כלל הנתונים זמינים לסינון לפי שנת פעילות (2023–2025) ומאפשרים בחינת מגמות, שינויים בהתנהגות שוק, וקבלת החלטות מבוססות עבר.

ה-VIEWS שמימשנו עבור חלק זה:

איגוד נתונים על מוצרים בכל אחת משלושת השנים האחרונות (2023-2025):

CREATE VIEW Product\_Annual\_Stats AS

SELECT

P.Product\_ID,

P.Name,

Y.Year,

-- Total Searches

ISNULL((

SELECT COUNT(\*)

FROM SEARCHES\_FOR\_PRODUCTS S

WHERE S.Product\_ID = P.Product\_ID AND YEAR(S.DT) = Y.Year

), 0) AS Total\_Searches,

-- Total Orders

ISNULL((

SELECT COUNT(DISTINCT O1.Order\_ID)

FROM ORDERED O1

JOIN ORDERS OR1 ON O1.Order\_ID = OR1.Order\_ID

WHERE O1.Product\_ID = P.Product\_ID AND YEAR(OR1.Date) = Y.Year

), 0)

+

ISNULL((

SELECT COUNT(DISTINCT G1.Order\_ID)

FROM GLASSES\_ORDERED G1

JOIN ORDERS OR2 ON G1.Order\_ID = OR2.Order\_ID

WHERE G1.Product\_ID = P.Product\_ID AND YEAR(OR2.Date) = Y.Year

), 0) AS Total\_Orders,

-- Total Units Sold

ISNULL((

SELECT SUM(O2.Units)

FROM ORDERED O2

JOIN ORDERS OR3 ON O2.Order\_ID = OR3.Order\_ID

WHERE O2.Product\_ID = P.Product\_ID AND YEAR(OR3.Date) = Y.Year

), 0)

+

ISNULL((

SELECT SUM(G2.Units)

FROM GLASSES\_ORDERED G2

JOIN ORDERS OR4 ON G2.Order\_ID = OR4.Order\_ID

WHERE G2.Product\_ID = P.Product\_ID AND YEAR(OR4.Date) = Y.Year

), 0) AS Total\_Units\_Sold,

-- Rating

CAST((

SELECT AVG(CAST(R.Rating AS FLOAT))

FROM REVIEWS R

WHERE R.Product\_ID = P.Product\_ID AND YEAR(R.Date) = Y.Year

) AS DECIMAL(4,2)) AS Rating

FROM PRODUCTS P

CROSS JOIN (SELECT 2023 AS Year UNION ALL SELECT 2024 UNION ALL SELECT 2025) AS Y

דו"ח המציג עבור כל שנה את יעד הרווח שנקבע בתחילתה ואת סך הרווח בפועל:

CREATE VIEW Revenue\_By\_Year AS

SELECT

Year,

Total\_Revenue,

Revenue\_Target = ROUND(LAG(Total\_Revenue) OVER (ORDER BY Year) \* 1.3,

0)

FROM (

SELECT

YEAR([Date]) AS Year,

SUM(Total\_Amount) AS Total\_Revenue

FROM ORDERS

GROUP BY YEAR([Date])

) AS YearlyRevenue

## מטלה 4 (10%) – אופטימיזציה של שאילתות באמצעות Generative AI

**להלן קישור לשיחה המלאה עם ה-Database Course Guru GPT:**

<https://chatgpt.com/share/6851d959-a1f4-8008-a4df-7ce2c784e9cb>

**אופטימיזציה מס' 1:**

בחרנו בשאילתת ה-SELECT ללא קינון השנייה, אשר מציגה את 10 המוצרים הנמכרים ביותר בשנת 2025.  
הוספנו את האינדקסים הבאים:

CREATE INDEX IX\_Orders\_Date ON Orders([Date], Order\_ID);

CREATE INDEX IX\_Ordered\_Product ON Ordered(Product\_ID, Order\_ID, Units);

CREATE INDEX IX\_Glasses\_Ordered\_Product ON Glasses\_Ordered(Product\_ID, Order\_ID, Units);

השאילתה החדשה תיראה כך (הפלט יישאר כמובן כפי שהיה):

SELECT TOP 10 P.Name, SUM(X.Units) AS Amount

FROM Products AS P

INNER JOIN (

SELECT O.Product\_ID, O.Units

FROM Ordered O

INNER JOIN Orders Ord ON O.Order\_ID = Ord.Order\_ID

WHERE Ord.[Date] >= '2025-01-01' AND Ord.[Date] < '2026-01-01'

UNION ALL

SELECT G.Product\_ID, G.Units

FROM Glasses\_Ordered G

INNER JOIN Orders Ord ON G.Order\_ID = Ord.Order\_ID

WHERE Ord.[Date] >= '2025-01-01' AND Ord.[Date] < '2026-01-01'

) AS X ON P.Product\_ID = X.Product\_ID

GROUP BY P.Name

ORDER BY Amount DESC;

**אופטימיזציה מס' 2:**

בחרנו בפונקציה השנייה שלנו, המקבלת מזהה ייחודי של לקוח ומחזירה את כל היסטוריית ההזמנות שלו**.**

הוספנו את האינדקסים הבאים:

CREATE INDEX IX\_CreditCards\_Customer ON CREDIT\_CARDS(Customer\_ID, Card\_Number);

CREATE INDEX IX\_Orders\_CardNumber ON ORDERS(Card\_Number, Order\_ID, [Date]);

CREATE INDEX IX\_Ordered\_Order ON ORDERED(Order\_ID, Product\_ID, Units);

CREATE INDEX IX\_GlassesOrdered\_Order ON GLASSES\_ORDERED(Order\_ID, Product\_ID, Version, Units);

CREATE INDEX IX\_Selected ON SELECTED(Product\_ID, Version, Feature, Selection);

CREATE INDEX IX\_Customizations ON CUSTOMIZATIONS(Feature, Selection, Extra\_Price);

הפונקציה החדשה תיראה כך (נצטרך כמובן להסיר את הפונקציה המקורית):

DROP FUNCTION IF EXISTS GetCustomerHistory

CREATE FUNCTION GetCustomerHistory (@CustomerID INT)

RETURNS TABLE

AS

RETURN

(

WITH Customer\_Orders AS (

SELECT O.Order\_ID, O.[Date]

FROM ORDERS O

INNER JOIN CREDIT\_CARDS CC

ON O.Card\_Number = CC.Card\_Number

WHERE CC.Customer\_ID = @CustomerID

),

Version\_Prices AS (

SELECT

V.Product\_ID,

V.Version,

P.Price + ISNULL(SUM(C.Extra\_Price), 0) AS Total\_Price

FROM VERSIONS V

INNER JOIN BASE\_FRAMES BF

ON V.Product\_ID = BF.Product\_ID

INNER JOIN PRODUCTS P

ON BF.Product\_ID = P.Product\_ID

LEFT JOIN SELECTED S

ON V.Product\_ID = S.Product\_ID AND V.Version = S.Version

LEFT JOIN CUSTOMIZATIONS C

ON S.Feature = C.Feature AND S.Selection = C.Selection

GROUP BY V.Product\_ID, V.Version, P.Price

),

Ordered\_Summary AS (

SELECT

O.Order\_ID,

SUM(D.Units) AS Units\_Ordered,

SUM(D.Units \* P.Price) AS Cost\_Ordered

FROM Customer\_Orders O

INNER JOIN ORDERED D

ON O.Order\_ID = D.Order\_ID

INNER JOIN PRODUCTS P

ON D.Product\_ID = P.Product\_ID

GROUP BY O.Order\_ID

),

Glasses\_Summary AS (

SELECT

O.Order\_ID,

SUM(G.Units) AS Units\_Glasses,

SUM(G.Units \* VP.Total\_Price) AS Cost\_Glasses

FROM Customer\_Orders O

INNER JOIN GLASSES\_ORDERED G

ON O.Order\_ID = G.Order\_ID

INNER JOIN Version\_Prices VP

ON G.Product\_ID = VP.Product\_ID AND G.Version = VP.Version

GROUP BY O.Order\_ID

)

SELECT

CO.Order\_ID,

CO.[Date],

ISNULL(OS.Units\_Ordered, 0) + ISNULL(GS.Units\_Glasses, 0) AS Total\_Units,

ISNULL(OS.Cost\_Ordered, 0) + ISNULL(GS.Cost\_Glasses, 0) AS Total\_Cost

FROM Customer\_Orders CO

LEFT JOIN Ordered\_Summary OS

ON CO.Order\_ID = OS.Order\_ID

LEFT JOIN Glasses\_Summary GS

ON CO.Order\_ID = GS.Order\_ID

)

## פרק שני - בונוס

## מטלה 5 – יישום כלים נוספים בלימוד עצמי (עד 2 נושאים, 2 נק' לכל נושא)

## נושא ראשון - PIVOT

נרצה ליצור שאילתה אשר מסייעת לנו בפילוח מכירות כל סוגי המשקפיים לפי שנים - כדי להשוות מכירות בין 2023, 2024 ו-2025 עבור כל סוגי המשקפיים.

השימוש ב-PIVOT מוצדק כאן מאחר והוא מאפשר להפוך את נתוני המכירות (יחידות שנמכרו) לשנים שונות לעמודות, וכך להציג בקלות השוואה בין השנים עבור כל מוצר. הצגה זו נוחה יותר לניתוח חזותי, לדוחות ולבניית דשבורדים, מאחר שכל שורה מייצגת מוצר וכל עמודה שנה במקום מבנה טבלאי רגיל עם ריבוי שורות למוצר אחד.

השאילתה:

SELECT

P.Product\_ID,

P.Name,

ISNULL(Sales.[2023], 0) AS Units\_2023,

ISNULL(Sales.[2024], 0) AS Units\_2024,

ISNULL(Sales.[2025], 0) AS Units\_2025

FROM PRODUCTS P

JOIN (

SELECT \*

FROM (

SELECT

G.Product\_ID,

YEAR(O.Date) AS Sale\_Year,

G.Units

FROM GLASSES\_ORDERED G

JOIN ORDERS O ON G.Order\_ID = O.Order\_ID

) AS SourceTable

PIVOT (

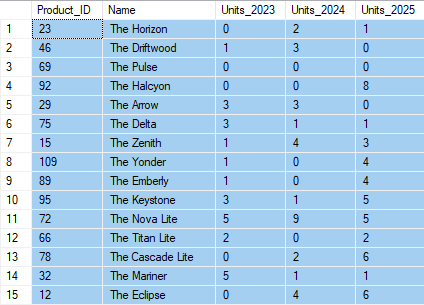
SUM(Units)

FOR Sale\_Year IN ([2023], [2024], [2025])

) AS PivotTable

) AS Sales ON P.Product\_ID = Sales.Product\_ID

פלט השאילתה (15 רשומות מתוך 88):



**נושא שני - TRY/ CATCH**

הפרוצדורה AddCustomerWithEmailValidation נועדה לוודא שהמשתמש מזין כתובת אימייל תקינה בעת ההרשמה. באמצעות מנגנון TRY...CATCH, היא מזהה שגיאות נפוצות כמו חוסר ב־@, נקודה, או סיומת לא מוכרת (לדוגמה .com, .org, .co.il), ומחזירה למשתמש הודעה ממוקדת וברורה. כל ניסיון כושל נרשם בטבלת לוג (EmailErrorLog), כולל פרטי האימייל והשגיאה. הלוגים האלו מאפשרים לארגון לנתח תבניות של טעויות משתמשים ולשפר את המערכת – לדוגמה, להציע תיקון אוטומטי לשגיאות שכיחות כמו הוספת @, או להנחות את המשתמש בזמן אמת. כך משפרים גם את חוויית המשתמש וגם את שיעור ההשלמה בפועל.

יצירת טבלת לוגים:

CREATE TABLE EmailErrorLog (

    LogID INT IDENTITY(1,1) PRIMARY KEY,

    EmailAddress NVARCHAR(255),

    ErrorDescription NVARCHAR(500),

    LogDate DATETIME DEFAULT GETDATE()

הפרוצדורה עצמה:

CREATE PROCEDURE AddCustomerWithEmailValidation

@FirstName NVARCHAR(100),

@LastName NVARCHAR(100),

@Email NVARCHAR(255)

AS

BEGIN

BEGIN TRY

-- @ בדיקת

IF CHARINDEX('@', @Email) = 0

BEGIN

THROW 51001, 'Invalid email address: missing @ symbol.', 1;

END

-- בדיקת נקודה

IF CHARINDEX('.', @Email) = 0

BEGIN

THROW 51002, 'Invalid email address: missing dot (.) symbol.', 1;

END

-- שליפת הסיומת

DECLARE @Suffix NVARCHAR(50);

SET @Suffix = LOWER(RIGHT(@Email, CHARINDEX('.', REVERSE(@Email)) - 1));

-- בדיקת סיומת חוקית

IF @Suffix NOT IN ('com', 'net', 'org', 'co.il', 'gov', 'edu')

BEGIN

THROW 51003, 'Invalid email domain suffix. Allowed: com, net, org, co.il, gov, edu.', 1;

END

-- הוספת הלקוח אם כל הבדיקות עברו

DECLARE @Customer\_ID INT;

SELECT @Customer\_ID = ISNULL(MAX(Customer\_ID), 0) + 1 FROM

CUSTOMERS;

INSERT INTO Customers (Customer\_ID, First\_Name, Last\_Name, Email)

VALUES (@Customer\_ID, @FirstName, @LastName, @Email);

PRINT 'Customer added successfully.';

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

INSERT INTO EmailErrorLog (EmailAddress, ErrorDescription)

VALUES (@Email, @ErrorMessage);

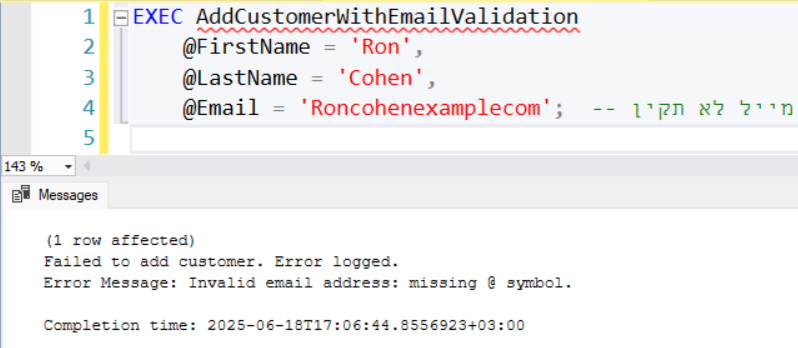
PRINT 'Failed to add customer. Error logged.';

PRINT 'Error Message: ' + @ErrorMessage;

END CATCH

END;

דוגמא להכנסת מייל לא תקין (ללא @):



דוגמא לטבלת ה-LOGS:

