

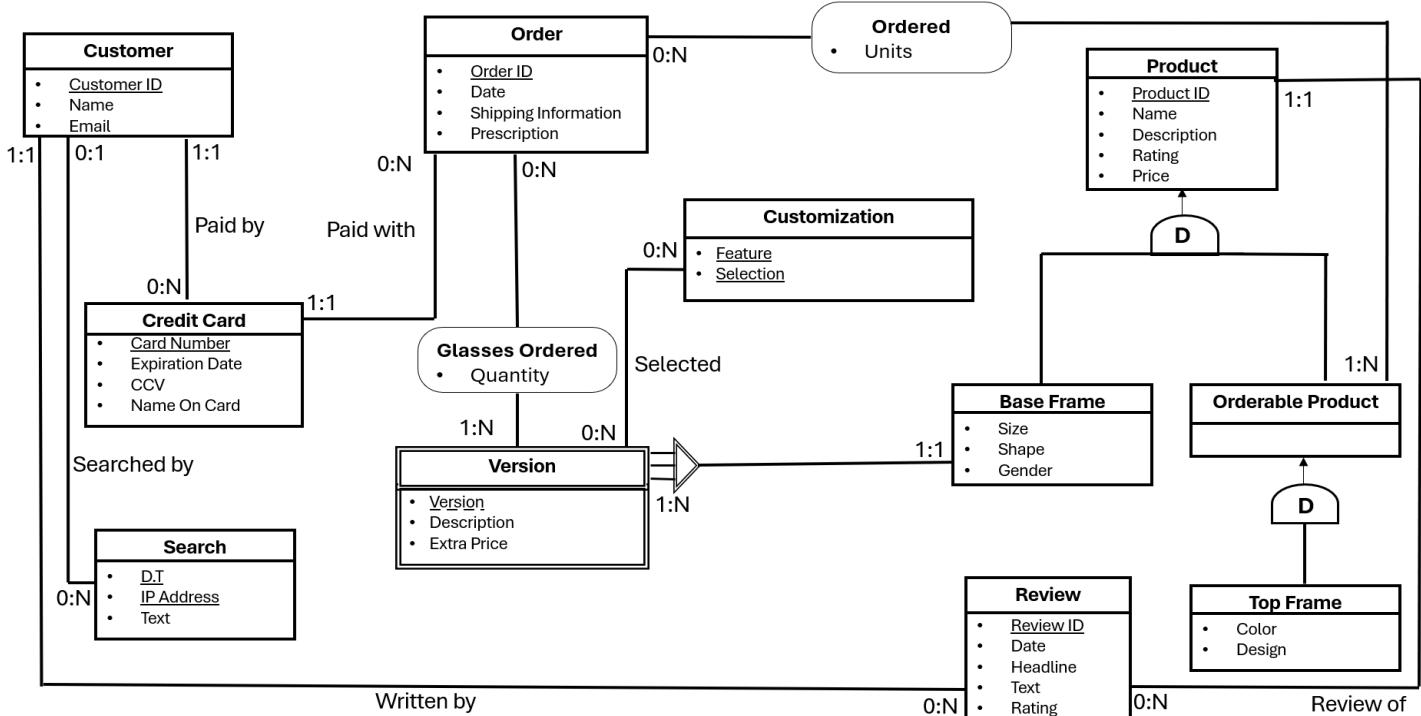
תאריך הגשה	אתר	מספר קבוצה
19/06/2025	www.paireyewear.com	15

פרויקט בסיסי נתונים – חלק ג'

פרק ראשון – מטלות חובה

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

מודל ERD שהוגש בחלק ב'





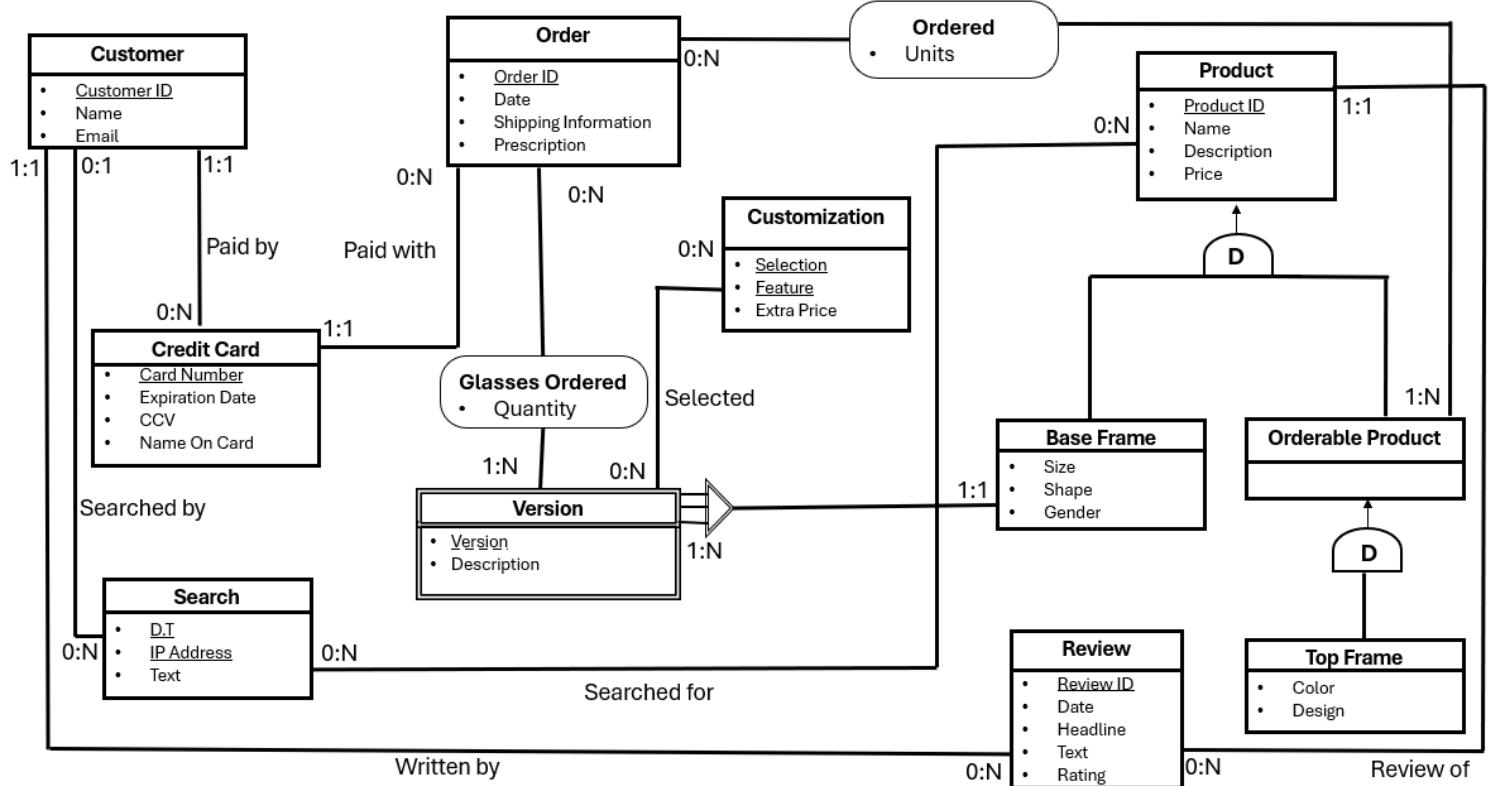
מודל טבלאי שהוגש בחלק ב'

- **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 ב מוצר הוא שדה מחושב והוא הממוצע של כל הדירוגים של אותו מוצר.
- התוספת מחיר צריכה להיות ב customization כי עבור כל התאמת יש מחיר אחר. איר תדעו את התוספת עברור כולם?
- צריך להיות קשר בין חיפוש למוצר שה מוצר הוא מה שוחר ב חיפוש.

מודל ERD מຕוקן



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))



מטרה 1 (35%) – שאלות

שתי שאלות SELECT ללא קנון (2.5%, 5% לכל שאלה)

שאלה מס' 1:

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

שאלת זו יכולה לסייע לעסוק בזיהוי לקוחות רוחניים במיוחד אשר רוכשים משקפיים בסכומים גבוהים ובכך לסייע בהצעת הטרופות למועדון VIP, קשר אישי למונעט נטישה למתחרים ודומה.

השאלה:

```
SELECT DISTINCT C.Customer_ID, SUM(P.Price*G_0.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_0 ON G_0.Order_ID=O.Order_ID
JOIN PRODUCTS P ON G_0.Product_ID=P.Product_ID
JOIN VERSIONS V ON V.Product_ID=P.Product_ID AND
G_0.[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_0.Units*P.Price)>2000
ORDER BY TC
```



פלט השאלה:

	Customer_ID	TC
1	110	2336.00
2	56	2349.00
3	83	2435.00
4	86	2456.00
5	52	2560.00
6	79	2583.00
7	31	2607.00
8	41	2648.00
9	78	2712.00
10	82	2755.00
11	95	2795.00
12	18	2864.00
13	49	3144.00
14	90	3146.00
15	55	3297.00
16	96	3581.00



שאילה מס' 2:

מי הם 10 המוצרים הכי נרכשים באתר מתחילת שנת 2025?

שאילתת זו יכולה לעזור לאבחן מוצרים פופולריים שלקוחות נטו לרכוש השנה האחרונות ולהציג אותם ללקוחות שטרם רכשו אותם.

השאילה:

```
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 השאילה:

	Name	Amount
1	The Starling	61
2	Glasses Travel Case	53
3	The Juniper	50
4	Lens Polishing Kit	48
5	Lens Protective Case	47
6	The Monarch	47
7	The Palisade	47
8	The Bellatrix	37
9	Glasses Cleaning Gel	34
10	Microfiber Cleaning	30



שתי שאלות SELECT מקוונות (% 10%, 5% לכל שאלה)

שאילה מס' 1:

בין כל המוצרים שבוצע עבורם חיפוש ע"י לקוחות, איזה מוצרים לא נרכשו לעולם?
שאילה זו יכולה לעזור לעסק לבחן מוצרים שלקוחות מעוניינים בהם ומחפשים אותם באתר, אך כאשר הלקוחות נכנסים לדף של מוצרים אלו - הם מתחרמים ובוחרים לא לרכוש אותם.

השאילה:

```
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 השאלה:

	Product_ID	Name
1	1	The Gemstone
2	2	The Black
3	3	The Navy
4	9	The Falcon
5	98	The Novara
6	99	The Obelisk
7	113	The Cirrus
8	115	The Elara
9	116	The Fenrir
10	118	The Helios
11	122	The Luminex
12	124	The Nebulae
13	126	The Peregrine
14	127	The Quicksilver
15	132	The Vesper



שאילה מס' 2:

מי הם הלוקוחות שדריגו מוצר כלשהו מתחת לדירוג הממוצע של המוצר?

שאילה זו עשויה לסייע בזיהוי הלוקוחות שדריגו מוצר באופן נמוך יחסית, באמצעות שאילתת זו ניתן לנתח את הביקורת שלהם ולבדוק האם ניתן לשפר את המוצר או חווית הרכישה

השאילה:

```
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 השאילה (מצגים 15 לקוחות מתוך 54 הלוקוחות שקיבלו בפלט):

	Customer_ID	Full Name
1	2	Quinn Martin
2	5	Hannah Thomas
3	8	Steve Davis
4	10	Michael Taylor
5	13	Charlie Garcia
6	18	Quinn Brown
7	33	Paula Rodriguez
8	39	Ian Taylor
9	43	Kevin Hernandez
10	47	Fiona Rodriguez
11	58	Nina Martinez
12	67	Ethan Brown
13	70	Ethan Gonzalez
14	74	Alice Moore
15	81	Fiona Anderson



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

שאלה מס' 1:

מהם הרביעונים הרוחים ביותר בכל שנה? בנוסף, כיצד השתנה הרוח של כל רביעון בהשוואה למקבילו בשנה הקודמת?

המטרה היא לנתח את ביצועי הרוחות בשנים האחרונות, ומשאלתא זו יכולה לנתח מידע על רביעונים יותר רוחיים, ולהשליך על אסטרטגיית השיווק והחשיפה של העסק בכל אחת מהתקופות.

בנוסף, נוכל לבחון האם אנחנו מוגמת עליית או ירידת, ולתת גם לנתח זה משקל בניתוח אסטרטגיות קודמות ותוכנו לכאה חדשות.

השאלה:

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 השאלה (15 רשומות מתוך 22 שהתקבלו):

	Year	Quarter	Quarterly_Revenue	Quarter_Rank	Same_Quarter_Last_Year	Revenue_Change
1	2025	1	34229.00	1	12081.00	22148.00
2	2025	2	27613.00	2	6722.00	20891.00
3	2024	1	12081.00	1	6340.00	5741.00
4	2024	3	12057.00	2	10278.00	1779.00
5	2024	4	9974.00	3	12426.00	-2452.00
6	2024	2	6722.00	4	19086.00	-12364.00
7	2023	2	19086.00	1	9065.00	10021.00
8	2023	4	12426.00	2	13089.00	-663.00
9	2023	3	10278.00	3	11244.00	-966.00
10	2023	1	6340.00	4	9241.00	-2901.00
11	2022	4	13089.00	1	14260.00	-1171.00
12	2022	3	11244.00	2	13011.00	-1767.00
13	2022	1	9241.00	3	5466.00	3775.00
14	2022	2	9065.00	4	9755.00	-690.00
15	2021	4	14260.00	1	13124.00	1136.00

שאילתת מס' 2:

מי הם הלוקחות שביבטו יחס גבוהה של הזמינות לעומת חיפושים (מחולק לרבעים), כאשר הרביע העליון יהיה 25% הלוקחות עם יחס זמן-חיפוש הגבוה ביותר) ומהי תדירות החיפוש הממוצעת שלהם?

מטרת השאלתה היא הצגת הלקחות המראים עניין ב מוצריו באתר אך אין מבצעים הרבה פעולות רכישה (ביחס לכמה שchimpו), לכן יהיה נכון להניע אותם לרכישה בדרךים מגוונות (מבצעים, תמריצים וכדומה). בנוסף השאלתת מצינית את תדירות החיפוש הממוצעת, תדירות זו יכולה ללמד על הרגלי החיפוש של הלקוח ולסייע בהטמת אסטרטגיות שיווקיות מותאמת זמן.

השאילה:

```

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 חיקום
                                               -- היחס בין מספר החיפושים להזמנתם
NTILE(4) OVER ( ORDER BY 1.0 * COUNT(DISTINCT S.DT) /
NULLIF(COUNT(DISTINCT O.Order_ID), 0) DESC ) AS Quartile, -- חלוקה ל-4 Quartiles
                                               -- חישוב ממוצע חיקום רכישה-חיפוש לפי Quartile
        ( SELECT AVG(DATEDIFF(DAY, Prev_Search_Date, Search_Date)) ) -- חישוב ממוצע חיקום רכישה-חיפוש בין
        הלוקות
    FROM (
        SELECT S1.DT AS Search_Date,
               LAG(S1.DT) OVER (PARTITION BY S1.Customer_ID ORDER BY S1.DT) AS
Prev_Search_Date -- קוחם של קודם הזמנה تاريخ
    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 השאלה (15 רשומות מתוך 63 סה"כ):

	Customer_ID	Full_Name	Number_Of_Searches	Number_Of_Orders	Ratio	Quartile	Avg_Days_Between_Searches
1	190	Kevin Smith	3	1	3.000	1	457
2	191	Diana Thomas	3	1	3.000	1	163
3	194	Bob Taylor	3	1	3.000	1	551
4	195	Oscar Martin	3	1	3.000	1	436
5	209	George Hernandez	3	1	3.000	1	296
6	208	George Anderson	2	1	2.000	1	761
7	200	Rachel Williams	2	1	2.000	1	3
8	201	Alice Garcia	2	1	2.000	1	2000
9	202	Kevin Rodriguez	2	1	2.000	1	168
10	203	Julia Taylor	2	1	2.000	1	1272
11	204	Michael Hernandez	2	1	2.000	1	1288
12	205	Ian Williams	2	1	2.000	1	727
13	206	Ian Martin	2	1	2.000	1	180
14	196	Ian Williams	2	1	2.000	1	777
15	198	George Martin	2	1	2.000	1	1423



שאילה מקוונת תוך שימוש ב-CTE (פסקת WITH) (10%)

מהן המדיניות הרוחניות ביותר עבור החברה? (בהתכלות על המדיניות עם: תדיות הזמןנות האנובוהה ביותר, כמוות הלקוחות האדולות ביותר, ההכנסות האנובוהות ביותר ועוד).

שאילתת זו תשיע לבחון את אסטרטגיית השיווק הנוכחית של העסק וכן עשויה לעזור בפתרון אסטרטגיה קידמה - היכן להשקיע יותר ואיפה פחות.

השאילה:

WITH

```
ORDER_INTERVALS AS (
    SELECT
        Country,
        Order_Date = 0.[Date],
        Prev_Order_Date = LAG(0.[Date]) OVER (PARTITION BY Country ORDER BY
        0.[Date]),
        Days_Since_Last_Order = DATEDIFF(DAY,
            LAG(0.[Date]) OVER (PARTITION BY Country ORDER BY 0.[Date]),
            0.[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
)
```



```
) 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 שהזרו סה"כ:

	Country	Avg_Days_Between_Orders	Active_Customers	Total_Orders	Total_Revenue	Revenue_Share	Efficiency
1	Germany	80.74	24	24	21952.00	0.0760	271.89
2	Argentina	74.36	28	29	17837.00	0.0617	239.88
3	USA	89.52	25	26	21248.00	0.0736	237.35
4	France	102.50	23	23	19970.00	0.0691	194.83
5	UK	86.00	20	20	11989.00	0.0415	139.41
6	Spain	120.46	14	14	13412.00	0.0464	111.34
7	Italy	163.71	15	15	17030.00	0.0590	104.02
8	Chile	107.81	22	22	11188.00	0.0387	103.78
9	Canada	143.94	17	17	11406.00	0.0395	79.24
10	Colombia	116.22	19	19	7876.00	0.0273	67.77



מטרה 2 (35%) – יישומי כלים מתקדמים

(5%) View

רשימת כל המוצרים, כאשר עבור כל מוצר מוצג: דירוג, מס' החיפושים שבוצעו עבورو, מס' ההזמנות בהן הזמיןו אותו, יחס חיפוש-הזמנה, מהי התדריות הממוצעת בין כל 2 הזמנות, מס' יחידות כולל שהוזמנו ממנו, מהי המדינה שהזמיןו ממנו הכי הרבה.

שאילתת זו תסייע בהציג פידבק של רוחבי וכלי עבור כל המוצרים באתר. לכן, טבעי שהעסוק יעשה שימוש בשאליתה זאת בתדריות גבוהה גם כשאליתה עצמאית וגם כתת-שאליתה עבור שאלות 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)
```

```
        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 (מצגים 10 רשומות מתוך 134):

```
SELECT *
FROM PRODUCTS_OVERVIEW
```

	Product_ID	Name	Rating	Total_Searches	Total_Orders	Search_Order_Ratio	Avg_Days_Between_Orders	Total_Units	Top_Country
1	1	The Gemstone	3.4	3	0	0	NULL	0	NULL
2	2	The Black	1	2	0	0	NULL	0	NULL
3	3	The Navy	3.67	2	0	0	NULL	0	NULL
4	4	Glasses Case	4	3	4	1.333	348.33	11	France
5	5	The Kirby	2.67	3	12	4	294.57	13	Canada
6	6	The Larkin	1	2	12	6	329.83	24	Germany
7	7	The Murphy	2	3	12	4	206.57	28	Italy
8	8	The Finley	3	3	12	4	323.83	22	Italy
9	9	The Falcon	3.5	2	0	0	NULL	0	NULL
10	10	The Orion	2	3	12	4	211.71	18	Spain



דוגמא לשימוש פרקי של ה-VIEW:

נרצה לענות על השאלה: מי הם עשרת המוצרים עם הדירוג הגבוה ביותר באתר, כמה העבודות בוצעו עבור כל אחד ממטופרים אלה ומה היא המדינה ממנה כל מוצר האמן 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

```

:VIEW-ה עם

```

SELECT TOP 10 Product_ID, Name, Rating, Total_Orders, Top_Country
FROM PRODUCTS_OVERVIEW
ORDER BY Rating DESC

```

פלט השאלה (כMOVED זהה עם וללא ה-W-)

	Product_ID	Name	Rating	Total_Orders	Top_Country
1	11	The Voyager	5	12	Spain
2	13	The Casual	5	12	Norway
3	23	The Horizon	5	12	Israel
4	32	The Mariner	5	12	Chile
5	41	The Luminary	5	9	Colombia
6	69	The Pulse	5	9	France
7	89	The Emberly	5	9	Australia
8	100	The Palisade	5	74	Argentina
9	104	The Talon	5	9	Netherlands
10	114	The Daggerfall	5	14	Italy

פונקציות (Functions) – 2 פונקציות – 5% לכל פונקציה)

פונקציה מס' 1:

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

הבראה 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 מוצריים באופן נמוך משמעותית מאשר הביקורות, או לחולופן באופן גבוה משמעותית מאשר הלקוחות.

דוגמא לשימוש פרקטי של הפונקציה: כתיבת TRIGGER שמריץ את הפונקציה עבור כל ביקורת שנכנסה לאתר. במידה ופלט הפונקציה הוא מס' שלילי שקטן מ민וס 3, סימן שהלקוח שכתב את הביקורת היה מושך לא מושך מה מוצר בהשוואה ללקוחות אחרים. מחלוקת שירות הלקוחות קיבל התראה על כך ותשלח מייל ללקוח שכתב את הביקורת.

***חשיבות ציון** – בפונקציה זו נבצע שימוש ב-WITH 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, עברו אילו ביקורות הפער בין הדירוג הנוכחי בвиוקורת לדירוג המוצר קטן מ민וס ??

```
SELECT *, Gap = dbo.GetRatingGap(Review_ID)  
FROM REVIEWS  
WHERE YEAR(Date) = YEAR(GetDate()) AND dbo.GetRatingGap(Review_ID) < -2
```

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

	Review_ID	Date	HeadLine	Text	Rating	Customer_ID	Product_ID	Gap
1	243	2025-07-23	Waste of money.	Seriously disappointed. It felt cheap and didn't las...	1	43	3	-2.67
2	300	2025-09-13	Poor quality.	This product didn't meet my expectations. The qu...	1	183	113	-2.22



פונקציה מס' 2:

פונקציה המתקבלת מזיהה ייחודי של לקוחות ומחייבת את כל ההיסטוריה הרצומות שלהם.

הצדקה להוספת הפונקציה:

הפונ' רלוונטיית עבור מקרים בהם נדרש לבדוק מייד על ההיסטוריה הרצומות של לקוחות מסוים. דוגמא לשימוש פרקטי של הפונקציה: לקוחות מתקשר לשירות לקוחות של האתר בטענה כי משLOW שהזמין לפני יותר מחודשים לא הגיע. אחת מהפעולות הראשונות שנצפה מנציג שירות לקוחות לבצע תהיה הפעלת הפונקציה כאשר הקלט הוא זיהוי הלקוח. במידה והפונקציה לא הייתה קיימת, זה יהיה צורך מנציג שירות לקוחות לכתוב שאלתה כדי לקבל את ההיסטוריה הרצומות של הלקוח וכתוצאה מכך לעצב את מתן השירות לקוחות.

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
    0.Order_ID,
    0.[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 הזמן עבור לקוח עם המזהה ?3

```

SELECT *
FROM dbo.GetCustomerHistory(3)

```

פלט השאלה:

	Order_ID	Date	Total_Units	Total_Cost
1	3	2021-11-09	6	697.00
2	238	2020-11-13	10	844.00



Trigger (10%)

יצרנו טריגר המעדכן מחיר של כל הזמנה לאחר הוספה או מחיקת מוצר כלשהו מהזמנה. מכיוון שיצלנו ישן 2 טבלאות אליהן נכנסות ההזמנות (טבלת ORDERED וטבלת ORDERED_GLASSES) עברו מוצרי משקפיים וטבלת ORDERED_GLASSES עברו מוצרי משקפיים) וניתן "להזין" רק לטבלה אחת, ביצענו 2 טריגרים, אחד עבור כל טבלה, אשר מעדכנים את סכום ההזמנה הכללי.

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 לטבלת ORDERS לערך האמתי 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
```

cut נכתוב טרייר לחישוב שדה Total_Amount כתוצאה מהוספה / מחיקה מ-ORDERED:

```
CREATE TRIGGER UpdateAmountOrderableProducts
```

```
ON ORDERED
```

```
AFTER INSERT, DELETE
```

```
AS
```

```
BEGIN
```

```
--  
רשומות הוספת ומחיקת מוצרים במידה
```

```
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
```

```
--  
רשומות מחיקת ומחיקת מוצרים במידה
```

```
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
    --רשומות הוספה ומתחזעת במידה
    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
```

--
רשומות מחייבת ומtbodyות במידה --

```
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:

לפחות בשם מichael martinez יוצר קשר עם שירות הליקוחות.
הוא מסביר שהזמין בטעות זוג משקפים בהתאם אישית מהازמנה האחרונה שלו - אך הוא אינו יודע את מזהה האזמנה. הוא כן יודע את כתובת האימייל שלו: michael.martinez14@outlook.org.
בנוסף, לאחר שתיאר מichael את זוג המשקפים לשירות הליקוחות, הנציג זיהה שמדובר ב:

.ProductID=14, Version=1

כעת, נציג השירות צריך למצוא את המזהה של האזמנה האחרונה של הליקוח לפי כתובת האימייל.
ואז למחוק ממנה את פרטי המשקפים הספציפיים.

לפני הרצת השאלה:

	Order_ID	Total_Amount
1	14	559.00

השאילה:

```
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 השאלה:

	Order_ID	Total_Amount
1	14	352.00

כפי שניתן לראות, הטריגר הופעל והשדה המחשב עודכן בהתאם.



프로그램 שומרה (10%) (Stored Procedure)

פרומפט שנייתן ל-ChatGPT:

שלום, אני צור פ्रוצדורה מאוחסנת המבוססת על הדרישות הבאות:
רקע: בכל שנה, ישם מספר אירועים ברחבי העולם שבהם רוב העסקים
מציעים הנחות משמעותיות באתר האינטרנט שלהם, כגון בלק פרידי, יום
הרווקים הסיני וכו'.

מטרה: עדכן ההנחה על מוצרים מסוימים לטווח תאריכים שיבחר
המשתמש ב-SP

קלטים: 1 - תקופת ההתחלה של ההנחה
EndDate@2@ - תקופת הסיום של ההנחה

LowerDiscount 3@-UpperDiscount 4@ - ההנחה הגבוהה ביותר שמוצר ברשימה יקבל
לוגיקה: בחזר מוצרים שעומדים בדרישות אלו: המוצרים שלא נמכרו ב-60
הימים האחרונים. ותן להם הנחה אקראית, אך ההנחה תהיה גבוהה יותר
כל עוד היחס בין דירוג למכירות גבוהה יותר.(שכל מוצר בעל דירוג טוב

יחסית אך לא נמכר הרבה ההנחה תהיה גבוהה יותר.)

פלטים: ה프וצדורה תעדכן את ההנחה בשדה discount של כל מוצר
שברשימה.



הקדמה לפרויקטורה - נוסיף שדה Discount:

```
ALTER TABLE PRODUCTS
```

```
ADD Discount Float
```

הפרויקטורה:

```
CREATE PROCEDURE ApplyDiscountsOnLowSellingProducts
```

```
@StartDate DATE,  
@EndDate DATE,  
@LowerDiscount FLOAT,  
@UpperDiscount FLOAT
```

```
AS
```

```
BEGIN
```

```
למכירות דירוג יחס לפי המוצרים כל של זמן עדכון --
```

```
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 *,
-- והטווה המקסימום את מהשבים
        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 -- מחלוקת להימנע
    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;
```

לפני הפעלת הפרוצדורה:
מצג 10 רשומות מתוך 134 רשומות (מוסרים)

	Product_ID	Name	Description	Price	Discount
1	1	The Gemstone	Top Frame in blue camo pattern.	25.00	NULL
2	2	The Black	A top frame for those who wants to stay classy.	30.00	NULL
3	3	The Navy	A top frame for true lovers of the sea.	28.00	NULL
4	4	Glasses Case	Our protective glasses case will make you feel safer.	20.00	NULL
5	5	The Kirby	This medium rectangle frame is stylish and timeless.	70.00	NULL
6	6	The Larkin	This narrow, modified rectangle frame is a fan-favo...	70.00	NULL
7	7	The Murphy	This wide frame has an oversized, new-fashioned ...	70.00	NULL
8	8	The Finley	This narrow frame is extremely versatile and effortl...	70.00	NULL
9	9	The Falcon	Blue-tinted top frame with a sleek design.	65.00	NULL
10	10	The Orion	A timeless frame with subtle detailing.	60.00	NULL

אחרי הפעלת הפרוצדורה:
מצג 20 רשומות מתוך 134 רשומות (모자רים)

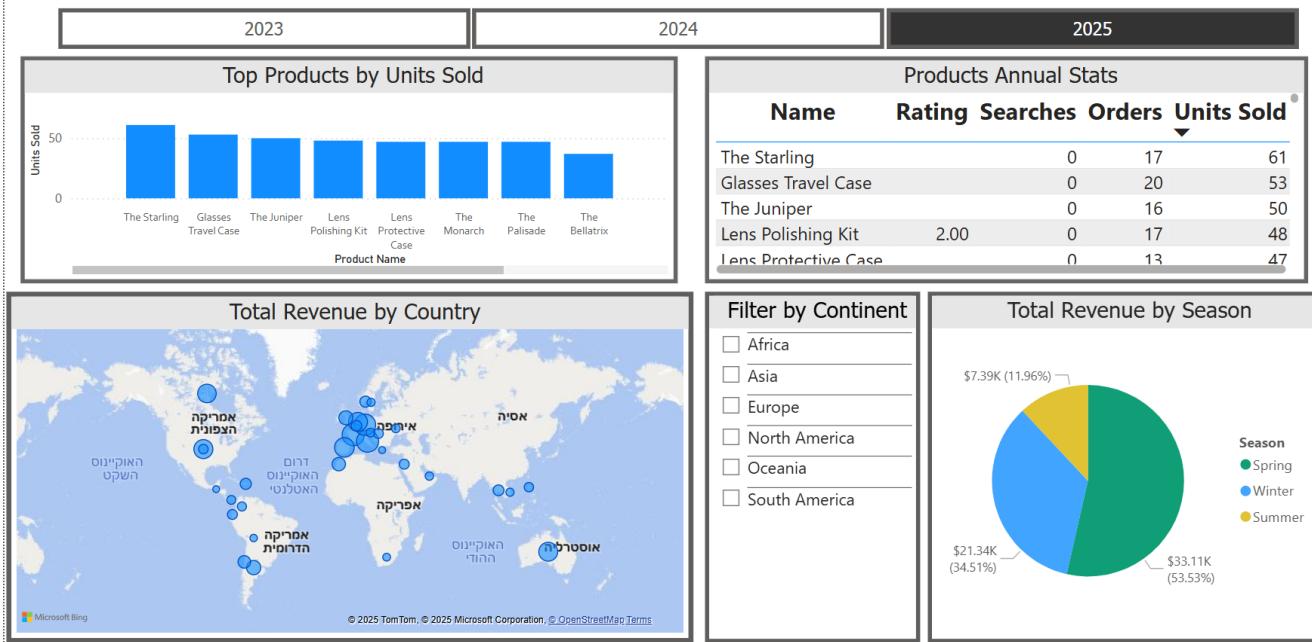
	Product_ID	Name	Description	Price	Discount
1	1	The Gemstone	Top Frame in blue camo pattern.	25.00	0.11
2	2	The Black	A top frame for those who wants to stay classy.	30.00	0.11
3	3	The Navy	A top frame for true lovers of the sea.	28.00	0.13
4	4	Glasses Case	Our protective glasses case will make you feel safer.	20.00	0.1
5	5	The Kirby	This medium rectangle frame is stylish and timeless.	70.00	0.31
6	6	The Larkin	This narrow, modified rectangle frame is a fan-favo...	70.00	0.18
7	7	The Murphy	This wide frame has an oversized, new-fashioned ...	70.00	0.26
8	8	The Finley	This narrow frame is extremely versatile and effortl...	70.00	0.34
9	9	The Falcon	Blue-tinted top frame with a sleek design.	65.00	0.14
10	10	The Orion	A timeless frame with subtle detailing.	60.00	0.26
11	11	The Voyager	Durable frame for the adventurous spirit.	75.00	0.5
12	12	The Eclipse	Modern frame with a matte finish.	70.00	0.31
13	13	The Casual	Lightweight and comfortable for daily wear.	55.00	0.5
14	14	The Aspen	Classic tortoise shell pattern.	58.00	0.45
15	15	The Zenith	Bold and sophisticated design.	62.00	0.42
16	16	The Atlas	Sturdy frame perfect for active users.	67.00	0.18
17	17	Lens Cleaning Kit	Complete cleaning kit for all glasses types.	15.00	NULL
18	18	Microfiber Cleaning	Soft microfiber cloth for scratch-free cleaning.	7.00	NULL
19	19	Anti-Fog Spray	Spray that prevents fogging on lenses.	12.00	NULL
20	20	The Riviera	Sleek metal frame with a modern touch.	68.00	0.26

מטלה 3 (20%) – כלים להציג נתונים

דו"ח עסקי למחלkat השיווק (10%)

נציג את הדוח'ה עסקי עבור מחלקה השיווק של חברת Pair Eyewear

Pair Eyewear - Marketing Report



הדו"ח נועד לנתח את דפוסי הרכישה באתר החברה, ומטרתו לספק למחלkat השיווק תובנות מבוססות נתונים לגבי העדפות של הלקוחות, זיהוי מגמות צרכניות, ו모יצרים בעלי פוטנציאל שיוקן שטרם מומש.

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



מרכיבי הדו"ח העסקי:

סר החזנות לפי מדינה עם פילוח ל匪בשות - מפה אינטראקטיבית המציגת המציג את מספר הרהמנות שביצעו בכל מדינה, עם אפשרות סינון לפי יבשת. תובנות אלו מסייעות למחלקת השיווק ליזמות אזוריים גאוגרפיים בעלי ביקוש גבוה ל מוצרים החברה, ולהתמקד בהם בكمפיינים שיווקיים ממוקדים לצורך הגדלת נתח השוק.

פילוח עונתי של הזמנות - תרשימים עוגה מציג את חלוקת ההזנות לפי עונות השנה (Winter, Spring, Summer, Autumn), במטרה להבין מגמות עונתיות בהעדפת הצרכנים. תובנה זו תורמת להתאמת ממוצי השיווק לפי תקופות שייא ושפלה לאורך השנה.

המוצרים הנמכרים ביותר - תרשימים עמודות מציג את עשרת המוצרים הנמכרים ביותר. ניתוח זה מאפשר למחלקת השיווק ליזמות מוצרים מוביילים, לקדם אותם באופן בולט, ולגבש תובנות לגבי העדפות הלוקחות.

טבלת כלל המוצרים ונוטני הביצועים שלהם - טבלה המציגת עבור כל מוצר את: הדירוג שלו, מס' החיפושים שביצעו עבורו, מס' ההזנות בהן הוא הוזן וסה"כ יחידות שנמכרו ממנו. בעקבות טבלה זו מחלקת השיווק תוכל ליזמות מוצרים עם פוטנציאל שטרם מומש או לחילופין מוצר שאין טעם להשקיע בו תקציב שיווקי.

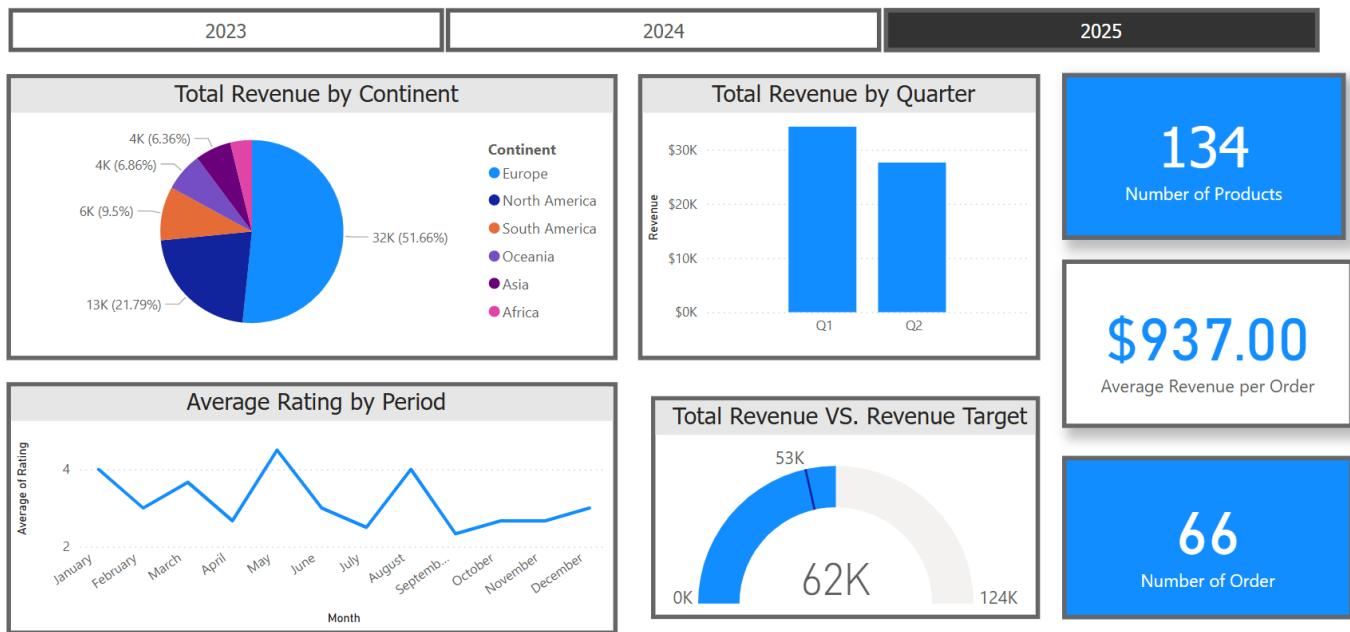
פילוח לפי שנה (2023-2025) - כלל המרכיבים בדוח ניתנים לפילוח דינמי לפי שנה באמצעות סלקטור, ובכךאפשרים לבחון מגמות משתנות לאורך זמן ולשפר את קבלת החלטות השיווקית בהתאם על נתונים עבר.

באמצעות תובנות אלו, מחלקת השיווק יכולה לבנות קMPIינים ממוקדים וمبוססי נתונים, לחדר את אסטרטגיית המכירה, ולהתאים את מגוון המוצרים וההיצע לצרכים המשתנים של הלוקחות ולמגמות השוק.

לוח מחוונים (10%)

מציג את הלוח המחוונים עבור הדרג הניהולי של חברת Pair Eyewear

Pair Eyewear - Management Dashboard



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.

*הברורה - תחת "Total Revenue By Quarter" מופיעים רק 1Q ו-2Q היוות ובחנו בפיתוח עבור שנות 2025 וכעת אנחנו נמצאים בסוף חודש יוני.



מרכיבי לוח המחוונים:

מוצע הדירוגים של ביקורות מוצרים באתר כפונקציה של הזמן - מدد המאפשר בחינת מגמות בשביות רצון הלkopות, עם אפשרות Down Drill לרמת רביעונים וחודשים. כלי זה מסיע ביזויו מוקדם של שינויים באיכות המוצר או חווית הלkop.

פיתוח רוח כולל לפי רביעונים - גرف המציג את הרוחים המمطلوبים ברבעונים, לצורך ניתוח עונתיות והבנת תקופותquia ביצועים העסקיים.

פיתוח רוח כולל לפי אוצר גאוגרפי - תרשימים עוגה המאפשר בחינה של רוח החברה לפי פילוח גיאוגרפי. יש תמיכה ב-Down Drill מרמת היבשות ועד רמת הערים.

השווות רוח בפועל מול רוח יעד - כל השוואתי המציג את הרוח הכלול לצד רוח היעד (הmbוסס על גידול של 30% מהשנה הקודמת). מאפשר ניטור עמידה ביעדים שונים ברמה ארגונית.

סה"כ מוצרים פעילים - אינדיקטיה למגון היצע שהחברה מספקת - מאפשר להבין את רוח הקטלוג של החברה ומסיע לביסוס קשר בין היצע לביצועים.

מוצע שווי להזמנה - ממד אסטרטגי לבחינת הרגלי הצריכה של הלkopות, התומך בהחלטות בנוגע למחיר, מבצעים ומסלולי מכירה.

סה"כ הזמניות - נתון הממחיש את היקף הפעולות הכלול של החברה.

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



ה-VIEWS שמיימשנו 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
```



מטרה 4 (10%) – אופטימיזציה של שאלות 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 בשאלת ה-SQL ללא קינון השנייה, אשר מוצגה 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);
```

השאלה החדשה תראה כך (הפלט יישאר כמוובן כפי שהיא):

```
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 לקוחות ומחזירה את כל ההיסטוריה הרצנותם 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);
```



הfonקציה החדשה תיראה כך (נצרך כموון להסיר את הfonקציה המקורי):

```
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 - בונים

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

נושא ראשון - PIVOT

נרצה ליצור שאלתה אשר מסיימת לנו בפיתוח מכירות כל סוגי השקפים לפי שנים - כדי להשוות מכירות בין 2023, 2024 ו-2025 עבור כל סוגי השקפים.

השימוש ב-PIVOT מוכיח כאן מאחר והוא מאפשר להפוך את נתוני המכירות (יחידות שנמכרו) לשנים שונות לעמודות, וכך להציג בקלות השוואת בין השנים עבור כל מוצר. הצעה זו נועה יותר לניתוח חזותי, לדוחות ולבניות דשبورדים, לאחר 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 השאלה (15 רשומות מתוך 88):

	Product_ID	Name	Units_2023	Units_2024	Units_2025
1	23	The Horizon	0	2	1
2	46	The Driftwood	1	3	0
3	69	The Pulse	0	0	0
4	92	The Halcyon	0	0	8
5	29	The Arrow	3	3	0
6	75	The Delta	3	1	1
7	15	The Zenith	1	4	3
8	109	The Yonder	1	0	4
9	89	The Emberly	1	0	4
10	95	The Keystone	3	1	5
11	72	The Nova Lite	5	9	5
12	66	The Titan Lite	2	0	2
13	78	The Cascade Lite	0	2	6
14	32	The Mariner	5	1	1
15	12	The Eclipse	0	4	6

נושא שני - TRY/ CATCH

הפרוצדורה AddCustomerWithEmailValidation נועדה לוודא שהמשתמש מזין כתובת אימייל תקינה בעת ההרשמה. באמצעות מנגןון TRY...CATCH, היא מזהה שגיאות נפוצות כמו חסר '@', נקודת, או סימנת לא מוכרת (לדוגמה .il, .com, .org, .co), ומחזירה למשתמש הודעה מוקדמת וברורה. כל ניסיון כשל נרשם בטבלת לוג (EmailErrorLog), כולל פרטי האימייל והשגיאה. הלוגים האלו מאפשרים לארಗן לנתח תבניות של טעויות משתמשים ולשפר את המערכת – לדוגמה, להציג תיקון אוטומטי לשגיאות שכיחות כמו הוספת '@', או להנחות את המשתמש בזמן אמת. כך מושנים גם את חווית 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

        -- שלילת הסיוםת --
        DECLARE @Suffix NVARCHAR(50);
        SET @Suffix = LOWER(RIGHT(@Email, CHARINDEX('. ', REVERSE(@Email)) -
    1));
```

```

-- בדיקת סיום 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

-- הוספה הלקוח אם כל בדיקות 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);

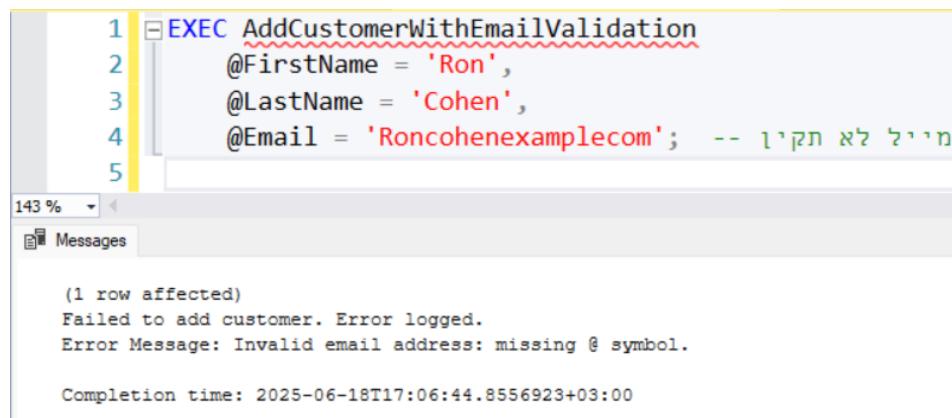
PRINT 'Customer added successfully.';
END TRY
BEGIN CATCH
DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();

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 @):



The screenshot shows a SQL Server Management Studio (SSMS) interface. In the top-left corner, there's a small icon of a person with a yellow background and a blue outline. The main area contains a T-SQL script:

```

1 EXEC AddCustomerWithEmailValidation
2     @FirstName = 'Ron',
3     @LastName = 'Cohen',
4     @Email = 'Roncohenexamplecom'; -- מייל לא תקין
5

```

In the bottom-right corner of the script window, there's a status message: "Query completed successfully." Below the script window, the "Messages" tab is open, showing the following output:

```

(1 row affected)
Failed to add customer. Error logged.
Error Message: Invalid email address: missing @ symbol.

Completion time: 2025-06-18T17:06:44.8556923+03:00

```



דוגמא לטבלת LOGS:

	LogID	EmailAddress	ErrorDescription	LogDate
1	5	Roncohen@examplecom	Invalid email address: missing @ symbol.	2025-06-18 17:17:10.237
2	6	david@gmail.com	Invalid email address: missing @ symbol.	2025-06-18 17:18:35.473
3	7	list@mydomain.xyz	Invalid email domain suffix. Allowed: com, net, ...	2025-06-18 17:18:48.997
4	8	noa@gmailcom	Invalid email address: missing dot(.) symbol.	2025-06-18 17:19:00.617