



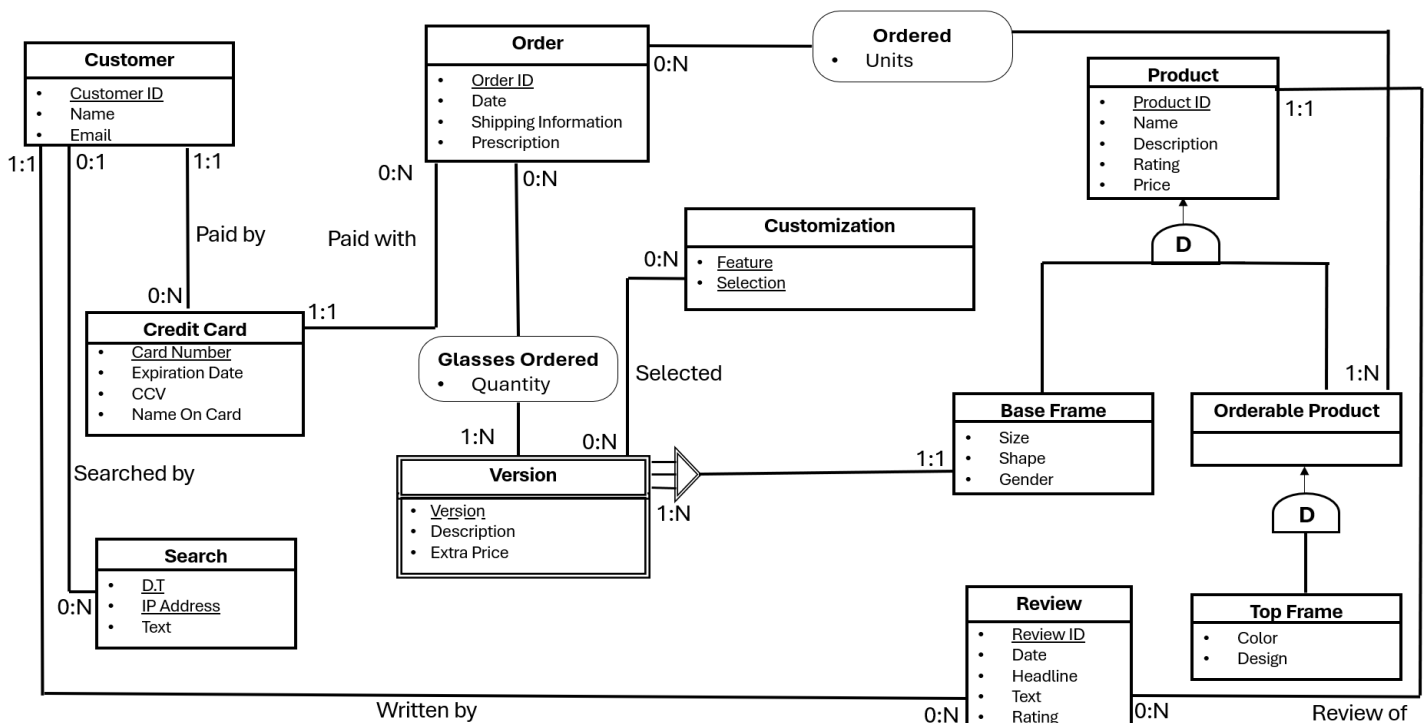
מס' קבוצה	אתר	תאריך הגשה
15	www.paireyewear.com	19/06/2025

## 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)**

### מודל ERD שהוגש בחלק ב'





## מודל טבלאי שהוגש בחלק ב'

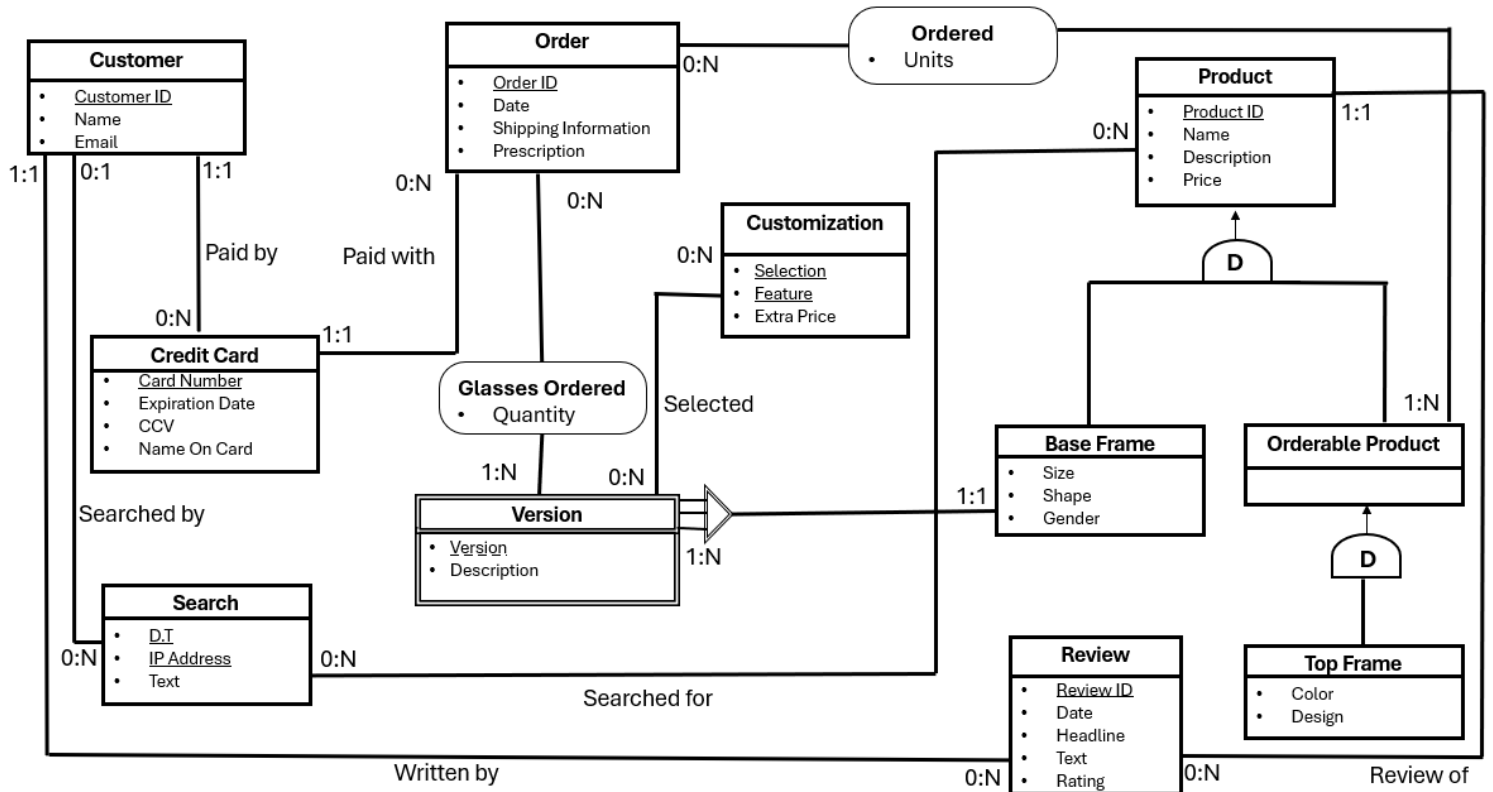
- **CUSTOMERS** ( Customer\_ID, First\_Name, Last\_Name, Email )
- **CREDIT\_CARDS** ( Card\_Number, Expiration\_Date, CCV, Name\_On\_Card, Customer\_ID (CUSTOMERS) )
- **SEARCHES** ( DT, IP\_Address, Text, Customer\_ID (CUSTOMERS) )
- **ORDERS** ( Order\_ID, Date, Country, City, Street\_Name, Street\_Number, 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%) – שאלות

### שתי שאלות SELECT ללא קינון (5%, 2.5% לכל שאלתה)

#### שאלתה מס' 1:

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

שאלתה 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.

#### השאלתה:

```
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 השאילתה:

	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



## שאלתה מס' 2:

מי הם 10 המוצרים הכי נרכשים באתר מתחילת שנת 2025?  
שאלתה זו יכולה לעזור לעסק לאבחן מוצרים פופולריים שלקוחות נטו לרכוש בשנה האחרונה ולהציע אותם ללקוחות שטרם רכשו אותם.

## השאלתה:

```
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 השאלתה:

	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 שאילות SELECT מקוננות (5%, 10% 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:

	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 Citrus
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





## שאלתה מס' 2:

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

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

## השאלתה:

```
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 השאלתה (מוציגים 15 לקוחות מתוך 54 הלקוחות שקיבלנו 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 (10%, 2.5% לכל שימוש נכון בפונקציה)

### שאלת 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, ומשאלתא 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.

### השאלתה:

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):

	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



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

## השאלתה:

```
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, -- היחס חישוב
           NTILE(4) OVER ( ORDER BY 1.0 * COUNT(DISTINCT S.DT) /
           NULLIF(COUNT(DISTINCT O.Order_ID), 0) DESC ) AS Quartile , -- הלקוחות חלוקת
           ( SELECT AVG(DATEDIFF(DAY, Prev_Search_Date, Search_Date)) -- ממוצע חישוב
           -- להזמנות למספר החיפושים מספר בין
           -- שלהם רכישה-חיפוש ליחס בהתאם רביעים 4 ל
           -- לקוח של קודם הזמנה תאריך --
           Prev_Search_Date
    FROM SEARCHES S1
    SELECT S1.DT AS Search_Date,
```



```

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):

	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



## שאלתה 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).

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

השאלתה:

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:

	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



## מטלה 2 (35%) – יישומי כלים מתקדמים

### View (5%)

רשימת כל המוצרים, כאשר עבור כל מוצר מוצג: דירוג, מס' החיפוש 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.

שאלתה זו תסייע בהצגת פידבק של רוחבי וכללי עבור כל המוצרים באתר. לכן, טבעי שהעסק יעשה שימוש בשאלתה זאת בתדירות גבוהה גם כשאלתה עצמאית וגם כתת-שאלתה 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,  
  
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 (מוציגים 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 פרקטי של ה-VIEW:

נרצה לענות על השאלתה: מי הם עשרת המוצרים עם הדירוג הגבוה ביותר באתר, כמה הזמנות בוצעו עבור כל אחד ממוצרים אלה ומה היא המדינה ממנה כל מוצר הזמין 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):

	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) (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):

	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 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:

	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 הוספת או מחיקת מוצר כלשהו מההזמנה. מכיוון שאצלנו ישנן 2 טבלאות אליהן נכנסות ההזמנות (טבלת ORDERED עבור מוצרים שאינם משקפיים וטבלת ORDERED\_GLASSES 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
    -- רשומות הוספת ומתבצעות 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 ומתבצעות 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 הוספת ומתבצעות 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 מחיקת ומתבצעות 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:

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:

	Order_ID	Total_Amount
1	14	559.00

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:

	Order_ID	Total_Amount
1	14	352.00

כפי שניתן לראות, הטריגר הופעל והשדה המחושב עודכן בהתאם.





## פרוצדורה שמורה (Stored Procedure) (10%)

פרומפט שניתן ל-ChatGPT:

שלום, אנא צור פרוצדורה מאוחסנת המבוססת על הדרישות הבאות:  
רקע: בכל שנה, ישנם מספר אירועים ברחבי העולם שבהם רוב העסקים מציעים הנחות משמעותיות באתרי האינטרנט שלהם, כגון בלק פריידי, יום הרווקים הסיני וכו'.  
מטרה: עדכון ההנחה על מוצרים מסוימים לטווח תאריכים שיבחר המשתמש ב-SP  
קלטים: 1 @StartDate - תקופת ההתחלה של ההנחה  
2 @EndDate - תקופת הסיום של ההנחה  
3 @LowerDiscount - ההנחה הנמוכה ביותר שמוצר ברשימה יקבל  
4 @UpperDiscount - ההנחה העליונה ביותר שמוצר ברשימה יקבל  
לוגיקה: בחר 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 גבוה יותר. (שככל שלמוצר בעל דירוג טוב יחסית אך לא נמכר הרבה ההנחה תהיה גבוהה יותר).  
פלטים: הפרוצדורה תעדכן את ההנחה בשדה discount של כל מוצר שברשימה.



הקדמה לפרוצדורה - נוסף שדה Discount:

```
ALTER TABLE PRODUCTS
```

```
ADD Discount Float
```

הפרוצדורה:

```
CREATE PROCEDURE ApplyDiscountsOnLowSellingProducts
```

```
@StartDate DATE,
```

```
@EndDate DATE,
```

```
@LowerDiscount FLOAT,
```

```
@UpperDiscount FLOAT
```

```
AS
```

```
BEGIN
```

```
-- למכירות דירוג יחס לפי המוצרים כל של זמני 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 *,

        -- והטווח המקסימום את מחשבים

        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 -- מחלוקה להימנע
ב0-

        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)

	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



אחריי הפעלת הפרוצדורה:

מוצג 20 רשומות מתוך 134 רשומות (מוצרים)

	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 Faloon	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

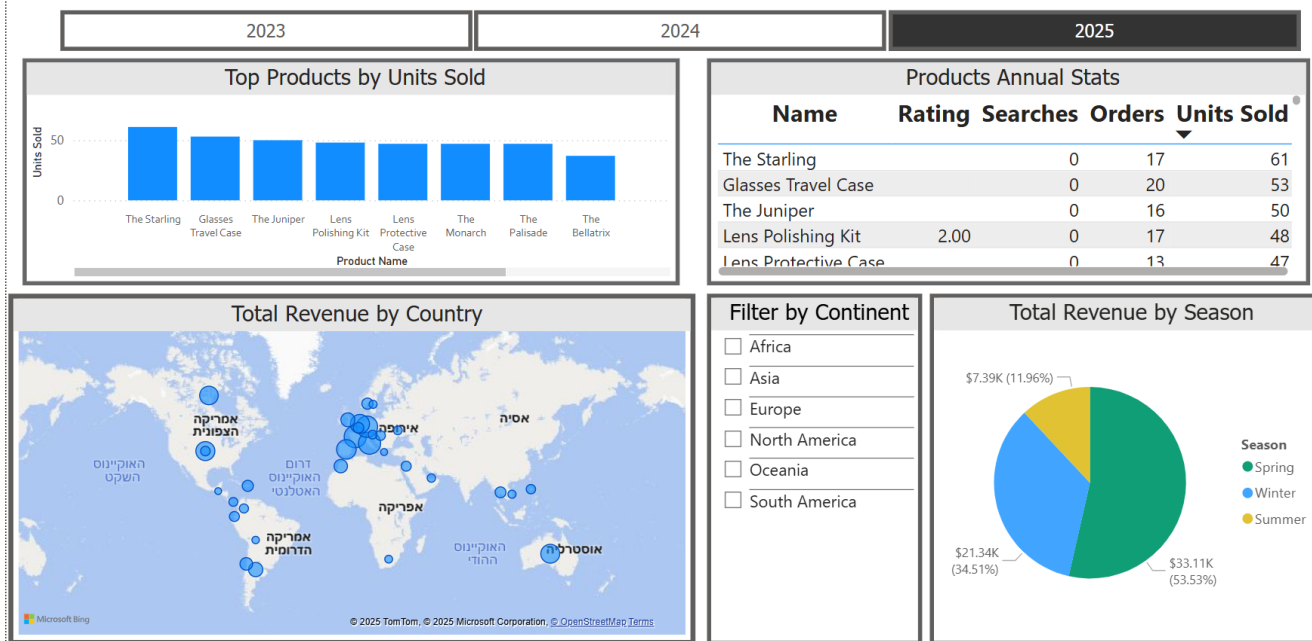


## מטלה 3 (20%) – כלים להצגת נתונים

### דו"ח עסקי למחלקת השיווק (10%)

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

## Pair Eyewear - Marketing Report



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.

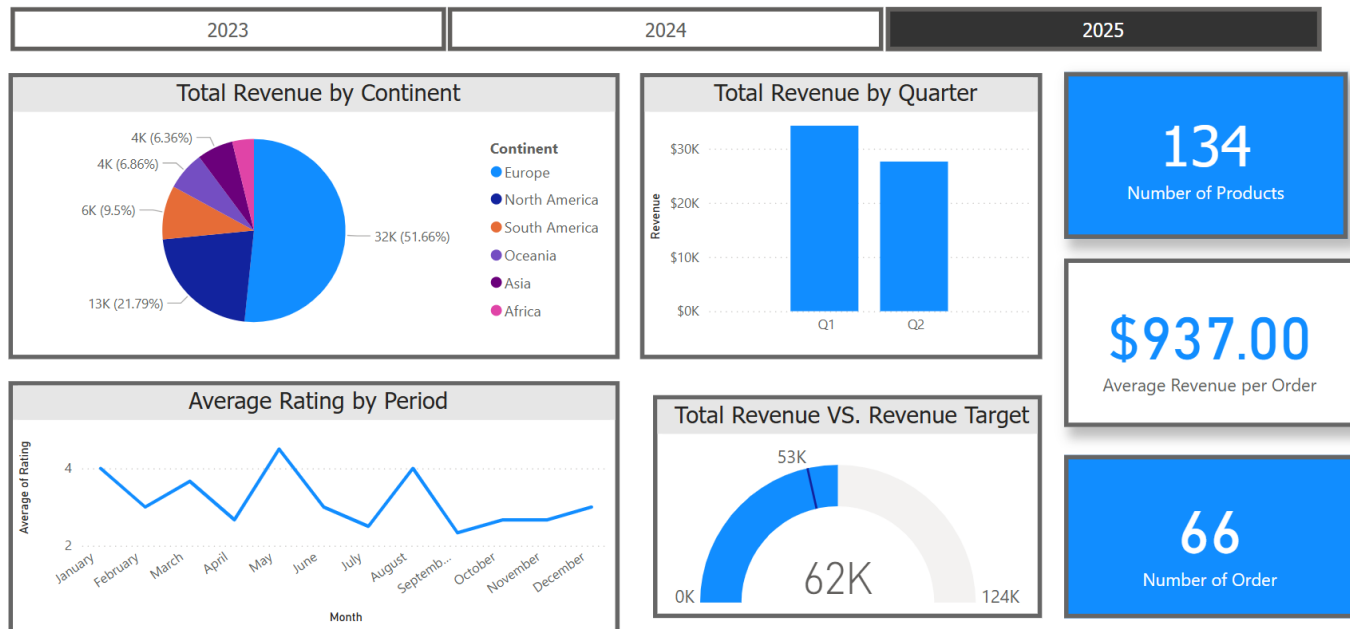
באמצעות תובנות אלו, מחלקת השיווק יכולה לבנות קמפיינים ממוקדים ומבוססי נתונים, לחדד את אסטרטגיות המכירה, ולהתאים את מגוון המוצרים וההיצע לצרכים המשתנים של הלקוחות ולמגמות השוק.



## לוח מחוונים (10%)

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

# 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.

\*הבהרה - תחת "Total Revenue By Quarter" מופיעים רק 1Q ו-2Q 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 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-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);
```

השאלתה 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):

	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 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

-- חוספת 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 @):

The screenshot shows a SQL Server query window with the following code:

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

Below the query window, the Messages pane displays 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 ה-LOGS:

	LogID	EmailAddress	ErrorDescription	LogDate
1	5	Roncohenexamplecom	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	liat@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