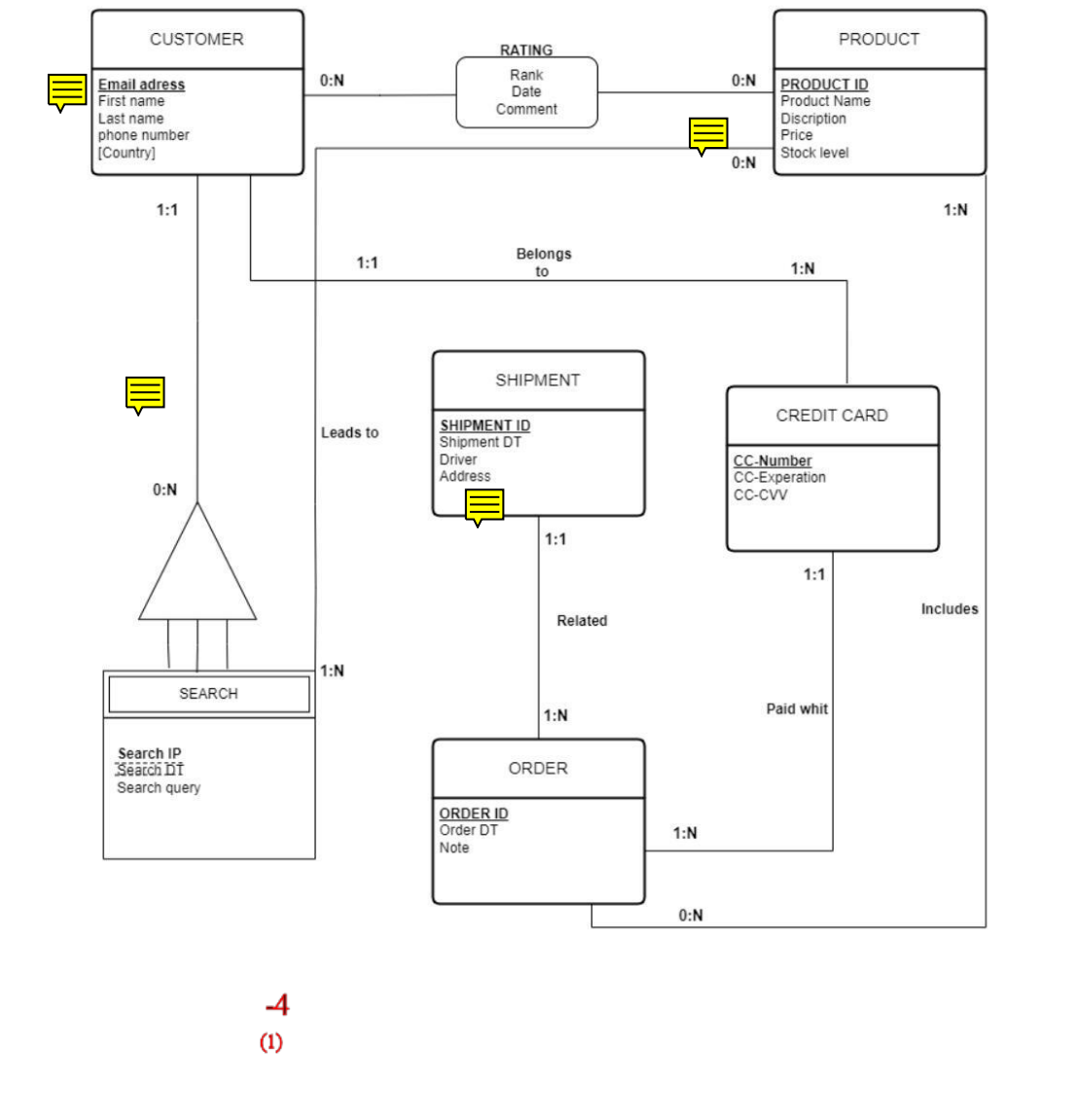
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מס' קבוצה | אתר | | | תאריך הגשה |
| 46 | <https://kustomkitgymequipment.com/> | | | 17/06/2023 |
| חברי הצוות - מספרי ת.ז | | | | |
| 322661075 | | 208343491 | 318720711 | |

## פרויקט בסיסי נתונים – חלק ג'

## מטלת חובה מקדימה – תיקון ה-ERD והעשרת בסיס נתונים

**ERD קודם:**

****

**הערות הבודק שצריך לתקן:**

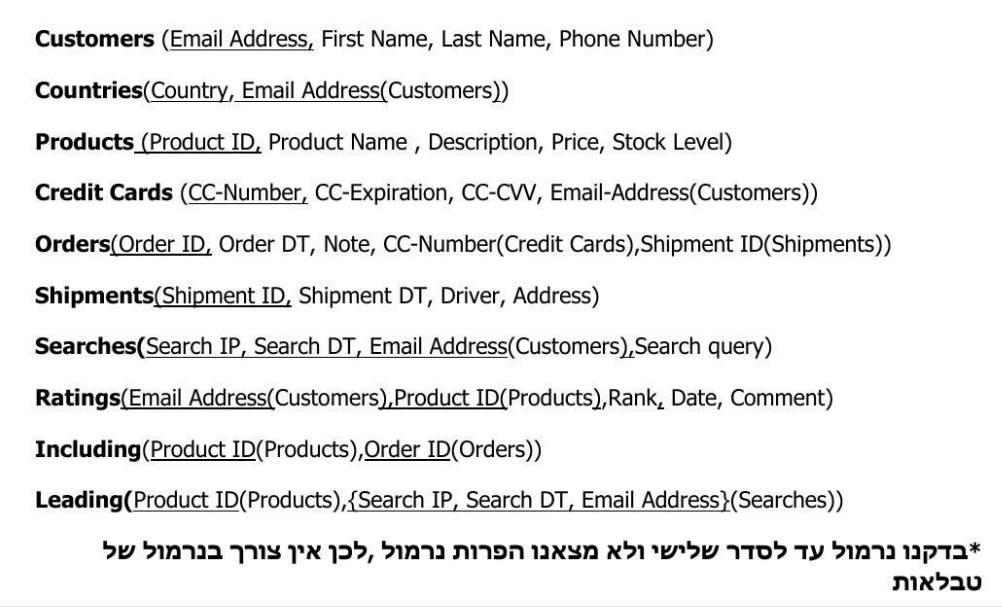
\* לא כל מוצר חייב להימצא באיזשהו חיפוש

\* איך זה הגיוני שלכל משלוח יש כמה הזמנות אבל רק כתובת אחת?

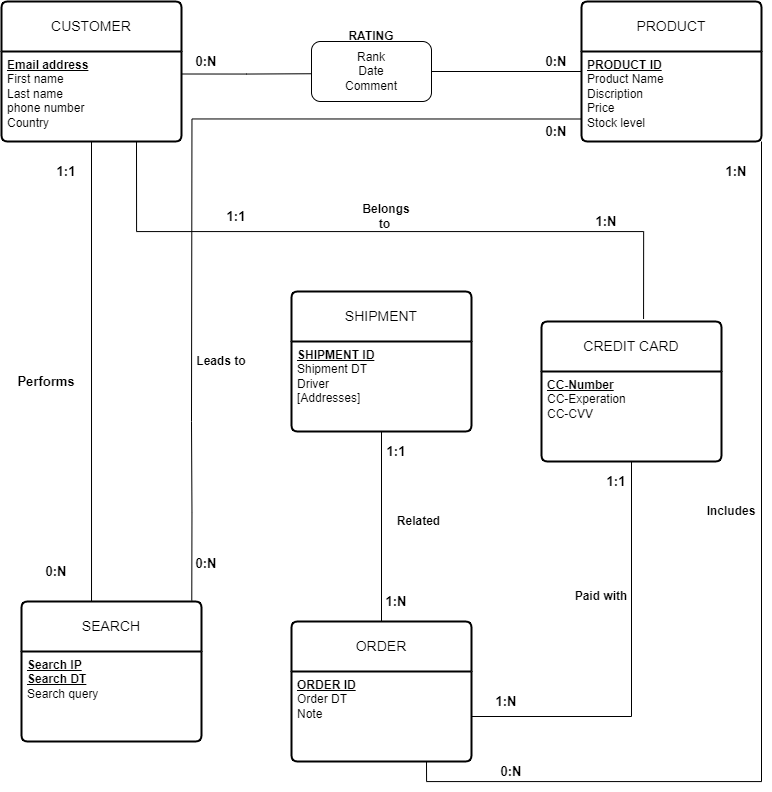
\* גם לקוח שלא רשום לאתר יכול לבצע חיפוש לכן לא הגיוני לעשות את החיפוש כישות חלשה של לקוח

\* למה country זה שדה מרובה ערכים?

**המודל הטבלאי בחלק ב':**

****

## תרשים ERD מתוקן:



**הנחות ERD מתוקן:**

|  |  |
| --- | --- |
| **מרכיב** | **הנחות** |
| RATING  קשר בין מוצר ללקוח | לקוח לא יכול לדרג מוצר מסוים יותר מפעם אחת. |
| CREDIT CARD | אמצעי תשלום אפשרי הינו אך ורק כרטיס אשראי אחד ויחיד. |
| SEARCH | לא בהכרח חיפוש יניב תוצאות. |
| הקשר בין לקוח להזמנה | ישנה אפשריות לבצע הזמנה מבלי תהליך חיפוש |

**מודל הטבלאי מתוקן:**

**Customers** (Email Address, First Name, Last Name, Phone Number,Country)

**Products** (Product ID, Product Name , Description, Price, Stock Level)

**Credit Cards** (CC-Number, CC-Expiration, CC-CVV, Email-Address(Customers))

**Orders**(Order ID, Order DT, Note, CC-Number(Credit Cards),Shipment ID(Shipments))

**Shipments**(Shipment ID, Shipment DT, Driver)

**Addresses**(Address, ShipmentID (Shipments))

**Searches(**Search IP, Search DT, Search query, Email Address(Customers))

**Ratings**(Email Address(Customers),Product ID(Products),Rank, Date, Comment)

**Including**(Product ID(Products),Order ID(Orders))

**Leading(**Product ID(Products),{Search IP, Search DT}(Searches))

\*בדקנו נרמול עד לסדר שלישי ולא מצאנו הפרות נרמול ,לכן אין צורך בנרמול של **טבלאות**

**קוד ה SQL המתוקן :**

**בניית הטבלאות לפי הסדר הנכון כולל אילוצי מרחב ערכים (אילוצי בדיקה (**

create table Addresses(

Address VARCHAR(50) Not Null primary key,

--lookup table for countries to help the user find the address

)

create table Customers(

EmailAddress Varchar(50) Not Null PRIMARY KEY,

FirstName Varchar(20) Not Null,

LastName Varchar(20) Not Null,

PhoneNumber VARCHAR(13)Not Null,

Country varchar(20) not null,

CONSTRAINT ck\_EmailAddress CHECK (EmailAddress LIKE '%@%.%'),

constraint CK\_phoneNumber CHECK (PhoneNumber LIKE '%[0-9]%')

--The first constraint checks for '@' and '.'

--The second constraint checks all digits as numbers

)

CREATE TABLE Products (

ProductID INT not null PRIMARY KEY,

ProductName VARCHAR(50) not null,

Description VARCHAR(100),

Price money,

StockLevel VARCHAR(3),

constraint CK\_Price CHECK (Price > 0 ),

constraint CK\_StockLevel CHECK (StockLevel IN ('yes','no'))

--The first constrain checks for price bigger than 0

--The second constraint checks for "yes" or "no" answer

)

CREATE TABLE Searches (

SearchIP INT not null,

SearchDT DATETime not null,

SearchQuery VARCHAR(50),

EmailAddress Varchar(50),

constraint CK\_Search primary key(SearchIP,SearchDT),

constraint fk\_EmailAddre foreign key (EmailAddress) REFERENCES Customers(EmailAddress)

)

CREATE TABLE Shipments (

ShipmentID INT not null PRIMARY KEY,

ShipmentDT DATE not null,

Driver VARCHAR(20),

Address VARCHAR(50),

CONSTRAINT fk\_Address foreign key (Address) REFERENCES Addresses(Address)

)

CREATE TABLE CreditCards (

CCNumber char(16) not null PRIMARY KEY,

CCExpiration Varchar(5) not null,

CCCVV CHAR(3)not null ,

EmailAddress Varchar(50) Not Null,

constraint fk\_Customer foreign key (EmailAddress) references Customers(EmailAddress),

constraint CK\_CCNumber CHECK (CCNumber LIKE '%[0-9]%'),

constraint CK\_CCExpiration CHECK (CCExpiration LIKE '[0-9][0-9]/[0-9][0-9]'),

constraint CK\_CCCVV CHECK (CCCVV LIKE '%[0-9]%')

--The first constraint checks for digits

--The second constraint checks for valid date

--The third constraint checks for digits

)

CREATE TABLE Orders (

OrderID INT not null PRIMARY KEY,

OrderDT DATE not null,

Note VARCHAR(50),

CCNumber char(16) not null,

ShipmentID INT not null,

constraint fk\_CreditCards foreign key (CCNumber) REFERENCES CreditCards(CCNumber),

constraint fk\_Shipment foreign key (ShipmentID) REFERENCES Shipments(ShipmentID),

)

CREATE TABLE Ratings (

EmailAddress VARCHAR(50) not null,

ProductID INT not null,

Rank INT,

RatingDate DATE,

Comment VARCHAR(100),

constraint Pk\_Rating PRIMARY KEY (EmailAddress, ProductID),

constraint fk\_Email foreign key (EmailAddress) REFERENCES Customers(EmailAddress),

constraint fk\_Product foreign key (ProductID) REFERENCES Products(ProductID),

constraint Ck\_Rank check (Rank between 0 and 5),

--The constrains checks for rank between 0 and 5

)

create table Including (

ProductID INT not null,

OrderID INT not null,

constraint Pk\_Includes PRIMARY KEY (OrderID, ProductID),

constraint fk\_Order foreign key (OrderID) REFERENCES Orders(OrderID),

constraint fk\_Product2 foreign key (ProductID) REFERENCES Products(ProductID),

)

create table Leading(

ProductID INT not null,

SearchIP INT not null,

SearchDT DATETime not null,

constraint Pk\_Leads PRIMARY KEY (ProductID, SearchIP,SearchDT),

constraint fk\_Search foreign key (SearchIP,SearchDT) REFERENCES Searches(SearchIP,SearchDT),

constraint fk\_Product3 foreign key (ProductID) REFERENCES Products(ProductID),

)

**מחיקת הטבלאות :**

--Delete script:--

DROP TABLE Leading

DROP TABLE Including

DROP TABLE Ratings

DROP TABLE Orders

DROP TABLE CreditCards

DROP TABLE Shipments

DROP TABLE Searches

DROP TABLE Products

DROP TABLE Customers

DROP TABLE Addresses

מטלה 1 (25%) – שאילתות

# שתי שאילתות SELECT ללא קינון (5%)

1. **שאילתה זו מחזירה מייל הלקוח, המדינה, כמות המוצרים שדירג, מחיר כולל של המוצרים ,עבור לקוח שדירג מעל מוצר אחד והדירוג של המוצרים מעל 3, בסדר יורד לפי כמות המוצרים שדירגו.**

SELECT C.EmailAddress, C.Country, Amount= count(P.ProductID), Total= SUM(P.Price)

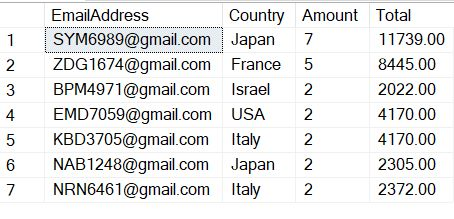
FROM Customers AS C join Ratings AS R on C.EmailAddress=R.EmailAddress join Products AS P on R.ProductID=P.ProductID

Where R.Rank>3

GROUP BY C.EmailAddress, C.Country

HAVING count(P.ProductID)>1

ORDER BY 3 DESC

****

1. **שאילתה זו מחזירה מזהה הזמנה, מזהה מוצר , תאריך הזמנה, נהג שספק ההזמנה, כתובת המשלוח, עבור הזמנות בשנת 2022, בסדר עולה לפי תאריך הזמנה**

SELECT O.OrderID,I.ProductID , O.OrderDT ,S.Driver, S.Address

FROM Orders AS O join Shipments AS S on O.ShipmentID= S.ShipmentID join Including as I on O.OrderID=I.OrderID

Where Year(O.OrderDT)=2022

ORDER BY 3 DESC**תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי**

# שתי שאילתות SELECT מקוננות (10%)

1. **שאילתה זו מחזירה מזהה מוצר, שם מוצר, מחיר חדש של המוצר אחרי הנחה 10%, עבור מוצר שנמצא במלאי ומחיר שלו גדול ממחיר הממוצע של כל המוצרים**

SELECT ProductID, ProductName, NewPrice= Price\*0.9

FROM Products

WHERE StockLevel in ('yes') and Price in (

SELECT Price FROM Products group by Price having price>(select avg(price) from Products))

ORDER BY 3 DESC

A picture containing text, screenshot, font, number

Description automatically generated

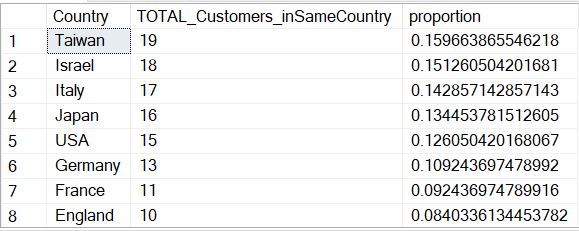
1. **שאילתה זו מחזירה מדינות, מספר הלקוחות בכל מדינה, הפרופורציה של לקוחות ששייכים למדינה מסוימת מתוך כל הלקוחות. מסודרות לפי מספר לקוחות במדינה באופן יורד.**

SELECT Country,TOTAL\_Customers\_inSameCountry = count (\*),proportion= Cast(Count(\*) as float)/Cast ((select count (\*) from Customers) as float)

FROM Customers

GROUP BY Country

order by 2 Desc



# שתי שאילתות מקוננות תוך שימוש במרכיבים נוספים (10%)

1. **שאילתה זו מוסיפה עמודה חדשה לטבלת המוצרים, ובתוך העמודה הזו נמצא ממוצע הדירוג לכל מוצר.**

ALTER TABLE Products ADD AvgRank int

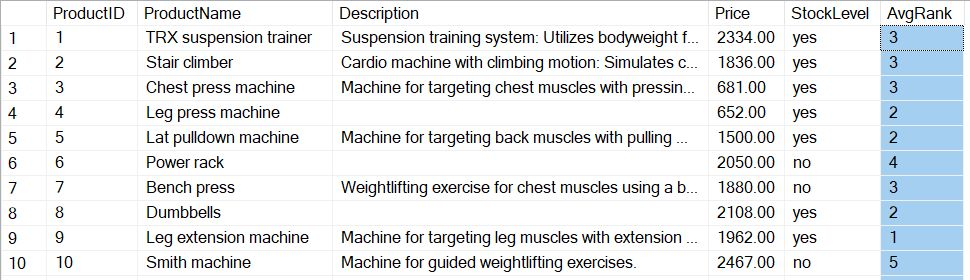
UPDATE Products

SET AvgRank = (

SELECT AVG(Rank)

FROM Ratings

WHERE Ratings.ProductID=Products.ProductID)



1. **שאילתה זו מחזירה מייל הלקוחות שעשו הזמנות שסכום שלהם מעל ממוצע מחירי כל המוצרים, אך לא דירגו אף מוצר.**

SELECT DISTINCT EmailAddress

FROM Customers

where EmailAddress IN (select EmailAddress

from Orders as O join CreditCards as Cr on O.CCNumber=Cr.CCNumber

join Including as I on O.OrderID=I .OrderID join Products as P on I.ProductID=P.ProductID

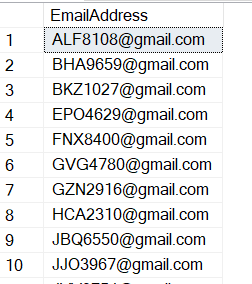
group by EmailAddress

having (select AVG(price) from products )<SUM(price))

Except

SELECT DISTINCT EmailAddress

FROM Ratings



## מטלה 2 (25%) – יישומי כלים מתקדמים

# View (5%).

**VIEW זה מציג מייל ומספר הזמנה של לקוחות פעילים(לקוח פעיל = לקוח שגם עשה הזמנה וגם דירג מוצרים)**

CREATE VIEW View\_ActiveCustomer AS

SELECT EmailAddress,O.OrderID

FROM Orders as O join CreditCards as Cr on O.CCNumber=Cr.CCNumber join Including as I on O.OrderID=I .OrderID join Products as P on I.ProductID=P.ProductID

where Cr.CCNumber in (select CCNumber from Orders) and EmailAddress in (select EmailAddress from Ratings)

group by EmailAddress,O.OrderID

**מימוש ה :VIEW**

**שאילתה זו מחזירה מייל , מדינה ,** **מספר הזמנות של לקוח שגם עשה הזמנה וגם דירג מוצרים(לקוח פעיל), עבור לקוח שעשה מעל 5 הזמנות. מסודרות לפי כמות ההזמנות באופן יורד.**

select distinct EmailAddress,Country=(select Country from Customers as C where C.EmailAddress=V.EmailAddress),TotalOrders=Count(V.OrderID)

from View\_ActiveCustomer as V

Group by EmailAddress

Having COUNT(V.OrderID)>5

Order by 3 DESC



# פונקציות (Functions) (10%)

1. **פונקציה סקאלרית זו מקבלת שם מוצר ומחזירה ממוצע הדירוג שלו**

CREATE FUNCTION RankOfPro ( @ProName varchar(50))

RETURNS int

AS BEGIN

DECLARE @Rankk Int

SELECT @Rankk = AVG (Rank)

FROM Products as P join Ratings as R On P.ProductID = R.ProductID

WHERE P.ProductName = @ProName

Group by ProductName

RETURN @Rankk

END

**מימוש הפונקציה :**

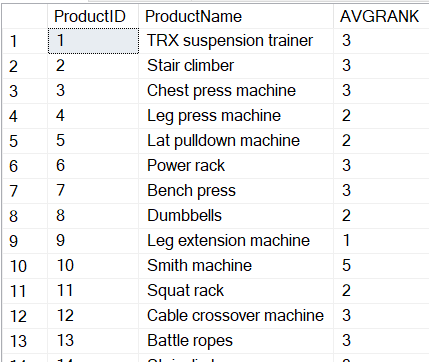
**למשל באופן** **כללי, שאילתה זו מחזירה מזהה מוצר, שם מוצר, ממוצע הדירוג של המוצר לפי שם המוצר:**

SELECT ProductID, ProductName,

AVGRANK = dbo.RankOfPro (ProductName)

FROM Products

ORDER BY ProductID

****

**למשל באופן ספיציפי אם פונקציה מקבלת שם מוצר Dumbbells :**

select AVGRank = dbo.RankOfPro('Dumbbells')

**תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי**

1. **פונקציה טבלאית זו מקבלת מזהה הזמנה ומחזירה טבלה שמפרטת את כלל שמי המוצרים שיש בה ומחיר שלהם.**

CREATE FUNCTION OrderDetails ( @OrdID int )

RETURNS TABLE

AS RETURN

SELECT OrderID , ProductName,Price

FROM Products as P JOIN Including as I On P.ProductID = I.ProductID

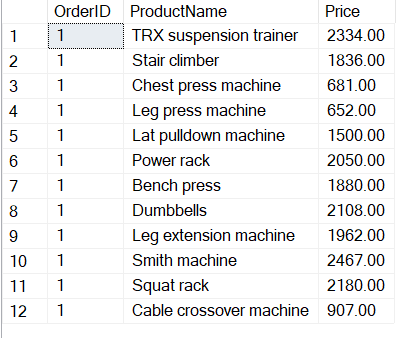
where OrderID = @OrdID

**מימוש הפונקציה :**

**למשל אם פונקציה מקבלת מזהה הזמנה "1" :**

Select \*

from dbo.OrderDetails(1)

****

# Trigger פשוט (5%)

**שאילתה זו מוסיפה עמודה חדשה לטבלת הלקוחות**

Alter table Customers Add LastSale Date

**הטריגר אומר שעבור כל הזמנה חדשה(אחרונה) ללקוח יתעדכן את תאריך ההזמנה הזה לעמודה החדשה שהוספנו בטבלת הלקוחות עבור אותו לקוח.**

Create TRIGGER update\_lastsale

ON Orders

for INSERT

AS

UPDATE Customers

SET LastSale = i.OrderDT

FROM Customers AS C

JOIN inserted AS i ON C.EmailAddress = (SELECT EmailAddress FROM CreditCards WHERE CCNumber = i.CCNumber)

**לפני ההזמנה החדשה:**

# 

**הזנת נתונים חדשים(הזמנה חדשה):**

# insert into Orders values (999,'7/7/2024','',2354439204744388,2)

**אחרי ההזמנה החדשה:**

\*\* מספר כרטיס אשראי **2354439204744388** שייך ללקוח בעל המייל **FNX8400@gmail.com**



# פרוצדורה שמורה ((Stored Procedure פשוטה (5%)

**פרוצדורה זו מקבלת דירוג מהלקוח ומחזירה לו את מזהה,שם,ממוצע דירוג לכל המוצרים שיש להם לפחות ממוצע דירוג כזה או יותר.**

CREATE PROCEDURE SP\_valuableProducts @rank int

AS

SELECT P.ProductID, ProductName,AVGRANK=AVG(Rank)

FROM Ratings as R join Products as P On R.ProductID = P.ProductID

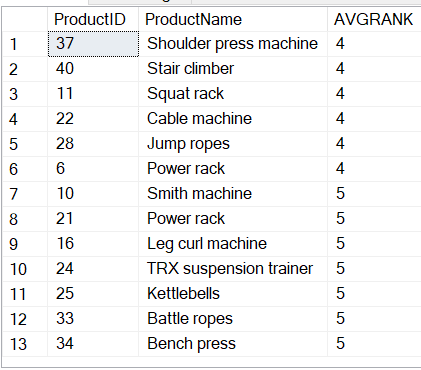
Group by P.ProductID, ProductName

having AVG (Rank) >= @rank

Order by AVG (Rank)

**הדגמת הפרוצדורה:**

EXECUTE SP\_valuableProducts  4



## מטלה 3 (20%) – כלים להצגת נתונים

**דו"ח עסקי (10%) –**

הדו"ח המוצג מיועד עבור מנהל מכירות, בעזרתו הוא יכול לקבל תובנות חשובות ולענות על שאלות שונות הקשורות למכירת מוצרים ודירוגי לקוחות.

**דירוג ממוצע לכל מוצר**

מנהל המכירות יכול להעריך את שביעות הרצון והתפיסה הכוללת של הלקוחות מכל מוצר

וכן אפשר לזהות מוצרים עם דירוגים ממוצעים גבוהים המציינים על איכות גבוהה.

**סך המכירות לכל מוצר מדורג**

מנהל המכירות יכול להשוות נתוני מכירות של מוצרים מדורגים לעומת מוצרים לא מדורגים.

דבר העוזר להעריך את ההשפעה של דירוג לקוחות על מכירות מוצרים.

**מציג סך מכירות לכל חודש**

מנהל המכירות יכול לנתח את ביצועי המכירות על בסיס חודשי.

אפשר לזהות מגמות ודפוסים עונתיים במכירות.

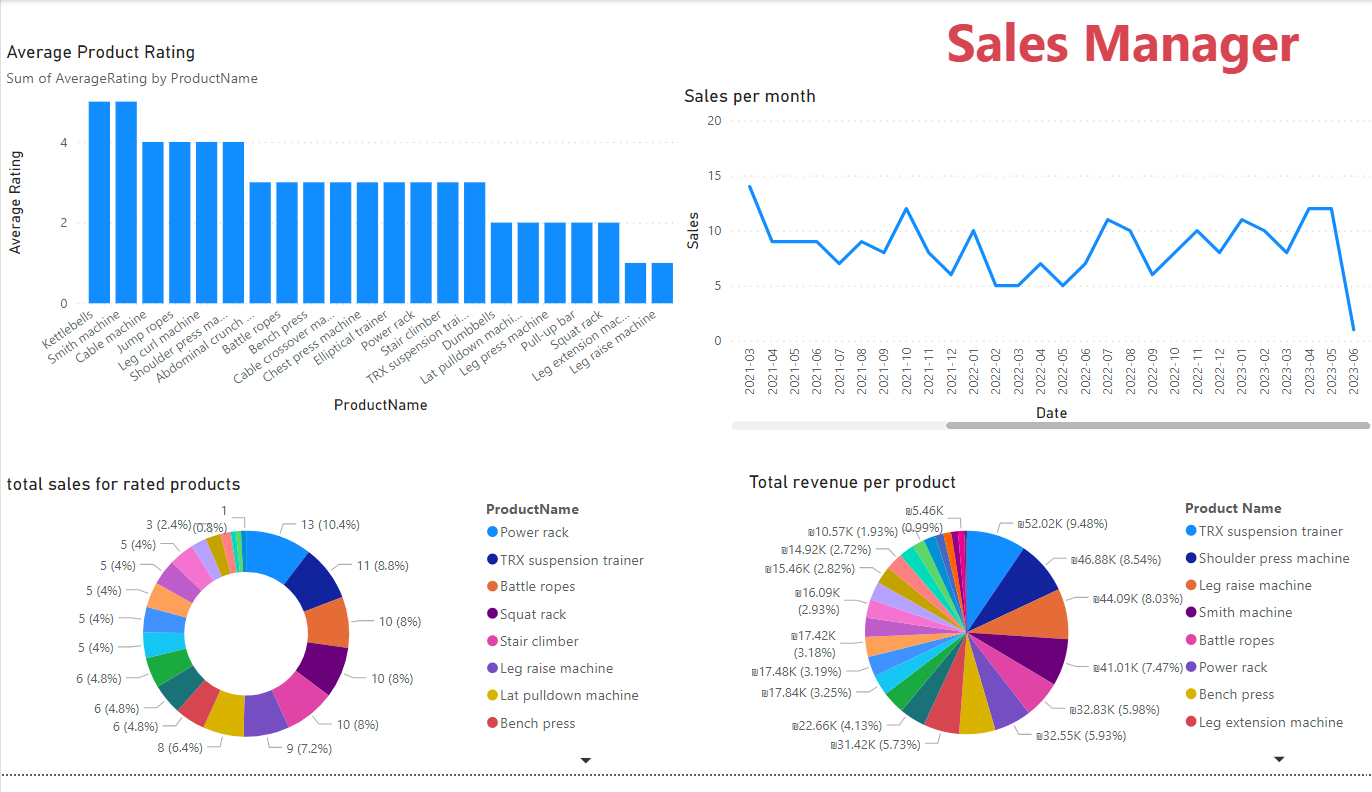
דבר היכול לעוזר בניטור תנודות מכירות ותכנון המלאי בהתאם.

**סך הכנסות מכל מוצר**

מנהל המכירות יכול לזהות את המוצרים הנמכרים ביותר על סמך סך ההכנסות שנוצרו.

ניתן להשוות את ההכנסות של מוצרים שונים כדי להעריך את הביצועים שלהם.

זה עוזר בהבנת התרומה הכוללת של כל מוצר להכנסות החברה.

****

**כדי לייצר את הדוח השתמשנו בשאילתות VIEW האלו :**

**זה מציג דירוג ממוצע לכל מוצר: VIEW**

-- Average ratings by product--

CREATE VIEW AverageRatingsByProduct AS

SELECT p.ProductName, AVG(r.Rank) AS AverageRating

FROM Products as p

JOIN Ratings as r ON p.ProductID = r.ProductID

GROUP BY p.ProductName

**זה מציג סך המכירות לכל מוצר מדורג: VIEW**

--Total sales with rating by product--

create VIEW TotalSalesByProduct AS

SELECT p.ProductName, COUNT(\*) AS TotalSales

FROM Products as p

JOIN Including as i ON p.ProductID = i.ProductID

where i.productID in (select productID from ratings)

GROUP BY p.ProductName

**זה מציג סך מכירות לכל חודש:VIEW**

--Total sales by month--

CREATE VIEW MonthlySales AS

SELECT SalesMonth, TotalSales

FROM (

SELECT FORMAT(OrderDT, 'yyyy-MM') AS SalesMonth, COUNT(\*) AS TotalSales

FROM Orders

GROUP BY FORMAT(OrderDT, 'yyyy-MM')

) AS Subquery

**זה מציג סך הכנסות מכל מוצר: VIEW**

--Total revenue by product--

CREATE VIEW TotalRevenueByProduct AS

SELECT p.ProductName, SUM(p.Price) AS TotalRevenue

FROM Products p

JOIN Including i ON p.ProductID = i.ProductID

GROUP BY p.ProductName

**לוח מחוונים (10%) -**

על ידי שימוש בתצוגה זו, מנהל החברה יכול לקבל תובנות לגבי יצירת הכנסות, התנהגות לקוחות, רמות מלאי וביצועי מוצרים. מידע זה מאפשר לו לקבל החלטות מושכלות כדי להניע את הצמיחה העסקית הכוללת, לייעל את המשאבים ולזהות תחומים לשיפור.

**סך הכנסות מכל מדינה**  
ניתן לנתח את ההכנסות שנוצרו ממדינות שונות שבהן הלקוחות נמצאים,

לזהות את המדינות עם הביצועים הטובים ביותר מבחינת הכנסות כדי להתמקד בהזדמנויות התרחבות. וכן להעריך את האפקטיביות של אסטרטגיות השיווק באזורים שונים.

**מציג את סך החיפושים בכל שעה**

ניתן להגיע לתובנות לגבי התנהגות החיפוש של הלקוחות וזיהוי שעות השיא של פעילות החיפוש,

לבצע אופטימיזציה של ביצועי האתר והקצאת משאבים עבור תמיכת לקוחות בשעות השיא של החיפוש. בנוסף להבין את העדפות על סמך מגמות חיפוש.

**כמות המוצרים שבמלאי וכמות המוצרים שלא במלאי**

ניתן לנהל מלאי ביעילות, לזהות מחסור במלאי ולהבטיח זמינות מוצרים כדי לענות על דרישת הלקוחות, לקבל החלטות מושכלות לגבי חידוש מלאי ואופטימיזציה של רמות המלאי.

**10 המוצרים בעלי ההכנסות הגדולות ביותר**

ניתן לראות את המוצרים בעלי הביצועים הטובים ביותר מבחינת הכנסות.

אפשר להקצות משאבים ולהתמקד בקידום מוצרים עתירי הכנסה אלו.

**רשימת מוצרים ללא מכירות ואשר לא במלאי**

ניתן להעריך את הביצועים והרלוונטיות של מוצרים אלה בשוק,

לקבל החלטות לגבי הפסקת מוצר, הוזלות פוטנציאליות או שינויים במערך המוצרים.

## 

**כדי לייצר את לוח מחוונים זה השתמשנו בשאילתות VIEW האלו :**

**זה מציג סך הכנסות מכל מדינה: VIEW**

--revenue by country--

CREATE VIEW RevenueByCountry AS

SELECT C.Country, SUM(P.Price) AS TotalRevenue

FROM Customers AS C

JOIN CreditCards AS Cr ON C.EmailAddress = Cr.EmailAddress

JOIN Orders AS O ON Cr.CCNumber = O.CCNumber

JOIN Including AS I ON O.OrderID = I.OrderID

JOIN Products AS P ON I.ProductID = P.ProductID

GROUP BY C.Country

**זה מציג את סך החיפושים בכל שעה: VIEW**

--Avg searches per hour--

CREATE VIEW SearchesPerHour AS

SELECT DATEPART(HOUR, SearchDT) AS SearchHour, COUNT(\*) AS TotalSearches

FROM Searches

GROUP BY DATEPART(HOUR, SearchDT)

**זה מציג את כמות המוצרים שבמלאי וכמות המוצרים שלא במלאי: VIEW**

--num of products in stock--

CREATE VIEW ProductsInStock AS

SELECT

SUM(CASE WHEN StockLevel = 'yes' THEN 1 ELSE 0 END) AS ProductsInStock,

SUM(CASE WHEN StockLevel = 'no' THEN 1 ELSE 0 END) AS ProductsOutOfStock

FROM Products

**זה מציג את 10 המוצרים בעלי ההכנסות הגדולות ביותר: VIEW**

--top 10 products based on total revenue--

CREATE VIEW Top10ProductsByRevenue AS

SELECT TOP 10 p.ProductName, COUNT(\*) AS TotalSales,

p.Price \* COUNT(\*) AS Revenue

FROM Products as p

JOIN Including i ON p.ProductID = i.ProductID

GROUP BY p.ProductName, p.Price

ORDER BY Revenue DESC

**זה מציג רשימת מוצרים ללא מכירות ואשר לא במלאי: VIEW**

--products without sales and without stock--

CREATE VIEW ProductsWithoutSales AS

SELECT p.ProductName, p.Price

FROM Products as p

LEFT JOIN Including i ON p.ProductID = i.ProductID

WHERE i.ProductID IS NULL and p.StockLevel = 'no'

## מטלה 4 (צבירה - עד 35%) – יישומים מתקדמים

## שאילתות עסקיות המשלבות Window Functions (10%)

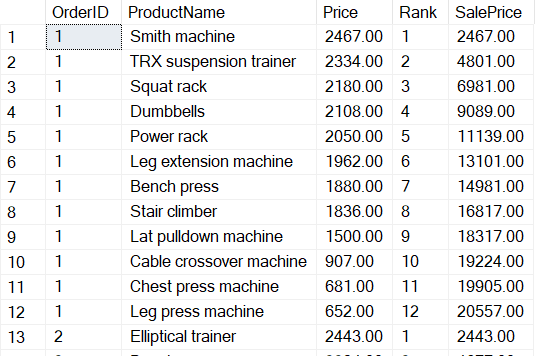
**1) שאילתת חלון זו מחזירה מזהה הזמנה,שם המוצרים ששיכים להזמנה,מחיר שלהם, הדירוג שלהם לפי המחיר מהיקר לזול ביותר, המחיר המצטבר.**

Select OrderID , ProductName , Price, Rank = Rank( ) over(Partition by OrderID order by Price desc ),

SalePrice = Sum(Price) over (partition by OrderID order by Price desc rows between UNBOUNDED preceding

and current row)

from Products as P JOIN Including as I On P.ProductID = I.ProductID



**2)שאילתת חלון זו מחזירה מזהה משלוח, תאריך משלוח, נהג המשלוח, כמות המשלוחים שעשה הנהג, מספר משלוח עבור הנהג לפי התאריך, ההפרש בימים בין תאריך כל משלוח לתאריך משלוח שלפניו, כמות ההזמנות במשלוח.**

SELECT S.ShipmentID, S.ShipmentDT, S.Driver,

DriverTotalShipment = COUNT(\*) OVER (PARTITION BY Driver),

ShipmentNumber=ROW\_NUMBER() OVER (PARTITION BY Driver ORDER BY ShipmentDT ),

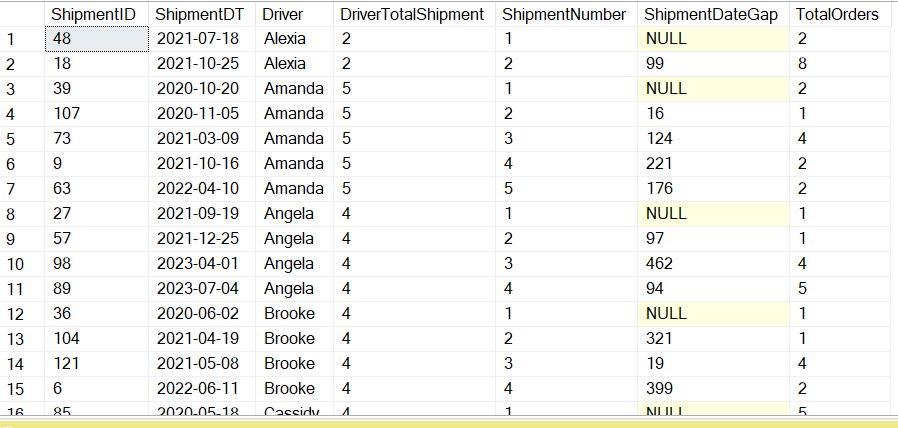
ShipmentDateGap = DATEDIFF(day, LAG(S.ShipmentDT) OVER (PARTITION BY Driver ORDER BY ShipmentDT), ShipmentDT),

TotalOrders=COUNT(O.OrderID)

FROM Shipments as S join Orders as O on S.ShipmentID=O.ShipmentID

group by S.ShipmentID, S.ShipmentDT, S.Driver

order by Driver



## שילוב מערכתי של מספר כלים (10%)

**שילוב מערכתי זה בודק אם כרטיס האשראי החדש של הלקוח(לקוח קיים) תקין או לא, אם תקין אז להוסיף אותו לטבלת כרטיסי אשראי ואם לא תקין אז לא להוסיף אותו( אחרי ההוספה ובדיקה שהוא לא תקין אז למחוק אותו מטבלה)**

**טריגר זה אומר שעבור כל הוספת כרטיס אשראי חדש, צריך להעביר את תוקף כרטיס האשראי( חודש ושנה) לפרוצדורה RemoveinValidCC**(**הטריגר מפעיל את הפרוצדורה)**

Create TRIGGER UpdateCreditCards

ON dbo.CreditCards

FOR INSERT AS

DECLARE @ExpM Varchar(5)

DECLARE @ExpY Varchar(5)

BEGIN

SELECT @ExpM = SUBSTRING(CCExpiration,1,2) , @ExpY =SUBSTRING(CCExpiration,4,2) FROM inserted

EXECUTE dbo.RemoveinValidCC @ExpM, @ExpY

END

**פרוצדורה זו מקבלת תוקף כרטיס האשראי ומעבירה אותו לפונקציה** IS\_CC\_VAILD **ואחרי הבדיקה של הפונקציה, במידה והוא פג תוקף (**'NotValid'**) אז מוחקת כרטיס האשראי מטבלה, אחרת (**'Valid'**) משאירה אותו(הפרוצדורה מפעילה את הפונקציה)**

Create PROCEDURE RemoveinValidCC (@ExpirationMonth Varchar(5), @ExpirationYear Varchar(5))

AS DECLARE @IsVaild varCHAR(8)

BEGIN

SET @IsVaild = dbo.IS\_CC\_VAILD(@ExpirationMonth, @ExpirationYear)

IF @IsVaild = 'NotValid'

DELETE FROM dbo.CreditCards WHERE CCExpiration= @ExpirationMonth + '/' + @ExpirationYear

END

**פונקציה זו מקבלת תוקף כרטיס האשראי ובודקת אותו, במידה והוא פג תוקף אז מחזירה**  'NotValid , **אחרת מחזירה** Valid

Create FUNCTION IS\_CC\_VAILD (@ExpMonth Varchar(5) ,@ExpYear Varchar(5))

RETURNS VarChar(8) AS

BEGIN

IF @ExpYear < substring(cast(YEAR(getdate()) as varchar(5)),3,2) OR (@ExpYear = substring(cast(YEAR(getdate()) as varchar(5)),3,2) AND @ExpMonth < MONTH(GETDATE()))

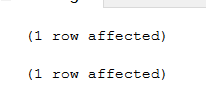
RETURN 'NotValid'

RETURN 'Valid'

END

**דוגמה עבור כרטיס אשראי פג תוקף:**

INSERT INTO CreditCards VALUES ('2000400050001000', '03/22', 444, 'ALF8108@gmail.com')

****

**ניתן לראות פה בגלל שהכרטיס פג תוקף נמחק אחרי ההסופה שלו(שתי פעולות).**

**דוגמה עבור כרטיס אשראי תקין:**

INSERT INTO CreditCards VALUES ('2000411115000100','10/24', 777,'ALF8108@gmail.com' )

**תמונה שמכילה גופן, לבן, טקסט, עיצוב

התיאור נוצר באופן אוטומטי**

**ניתן לראות פה בגלל שהכרטיס תקין התווסף לטבלה ונשאר.**

**תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי**

## דו"ח המושתת על שימוש בפסקת WITH מורכבת (10%)

**דוח עסקי זה מתמקד במגמות הקניות של לקוחות מובילים ממדינות מובילות ובמוצרים פופולאריים שמדורגים גבוהים, הדוח מציג מדינות המובילות שסכום ההזמנות של לקוחות ממדינות האלו מעל 40 הזמנות, כמות הזמנות של מדינות מובילות, סכום ההזמנות של מדינות מובילות, שמות הלקוחות המובילים (ששייכים למדינות המובילות) שביצעו יותר משתי הזמנות וסכום ההזמנות שלהם מעל 5000, כמות הזמנות של לקוחות המובילים, סכום הקניות של לקוחות המובילים, שמות המוצרים הפופולריים(שנקנו ע"י הלקוחות המובילים השייכים למדינות המובילות) שנמכרו מעל פעמיים, כמות המכירות של המוצרים הפופולאריים, הדירוג לכל מוצר פופולארי עבור לקוח מוביל שדירג אותו מעל 3. התוצאות ממוינות לפי שם המדינה.**

WITH TopCountries AS (

SELECT C.Country, COUNT(DISTINCT O.OrderID) AS TotalOrdersForCountry, SUM(P.Price) AS PurchasesAmountForCountry

FROM

Customers AS C

JOIN CreditCards AS CC ON C.EmailAddress = CC.EmailAddress

JOIN Orders AS O ON CC.CCNumber = O.CCNumber

JOIN Including AS I ON O.OrderID = I.OrderID

JOIN Products AS P ON I.ProductID = P.ProductID

GROUP BY C.Country

Having COUNT(DISTINCT O.OrderID)> 40

),

TopCustomers AS (

SELECT C.EmailAddress, CustomerName = C.FirstName + ' ' + C.LastName, C.Country,

COUNT(DISTINCT O.OrderID) AS TotalOrdersForCust,

SUM(P.Price) AS CustomerTotalSpent

FROM

Customers AS C

JOIN CreditCards AS CC ON C.EmailAddress = CC.EmailAddress

JOIN Orders AS O ON CC.CCNumber = O.CCNumber

JOIN Including AS I ON O.OrderID = I.OrderID

JOIN Products AS P ON I.ProductID = P.ProductID

GROUP BY C.EmailAddress, C.FirstName, C.LastName,C.Country

HAVING COUNT(DISTINCT O.OrderID) >= 3 and SUM(P.Price)> 5000

),

PopularProducts AS (

SELECT P.ProductID, P.ProductName, COUNT(I.OrderID) AS TotalOrdersForPro

FROM

Products AS P

JOIN Including AS I ON P.ProductID = I.ProductID

GROUP BY P.ProductID, P.ProductName

having COUNT(I.OrderID)>2

),

HighRatedProducts AS (

SELECT P.ProductID,R.EmailAddress, Rank AS HighRank

FROM

Products AS P

JOIN Ratings AS R ON P.ProductID = R.ProductID

where Rank > 3

GROUP BY P.ProductID,R.EmailAddress,Rank

)

SELECT

TP.Country,

TP.TotalOrdersForCountry,

TP.PurchasesAmountForCountry,

TC.CustomerName,

TC.TotalOrdersForCust,

TC.CustomerTotalSpent,

PP.ProductName,

PP.TotalOrdersForPro,

HR.HighRank

FROM

TopCountries AS TP

JOIN TopCustomers AS TC ON TP.Country = TC.Country

JOIN HighRatedProducts AS HR ON TC.EmailAddress = HR.EmailAddress

JOIN PopularProducts AS PP ON HR.ProductID = PP.ProductID

ORDER BY

TP.Country

