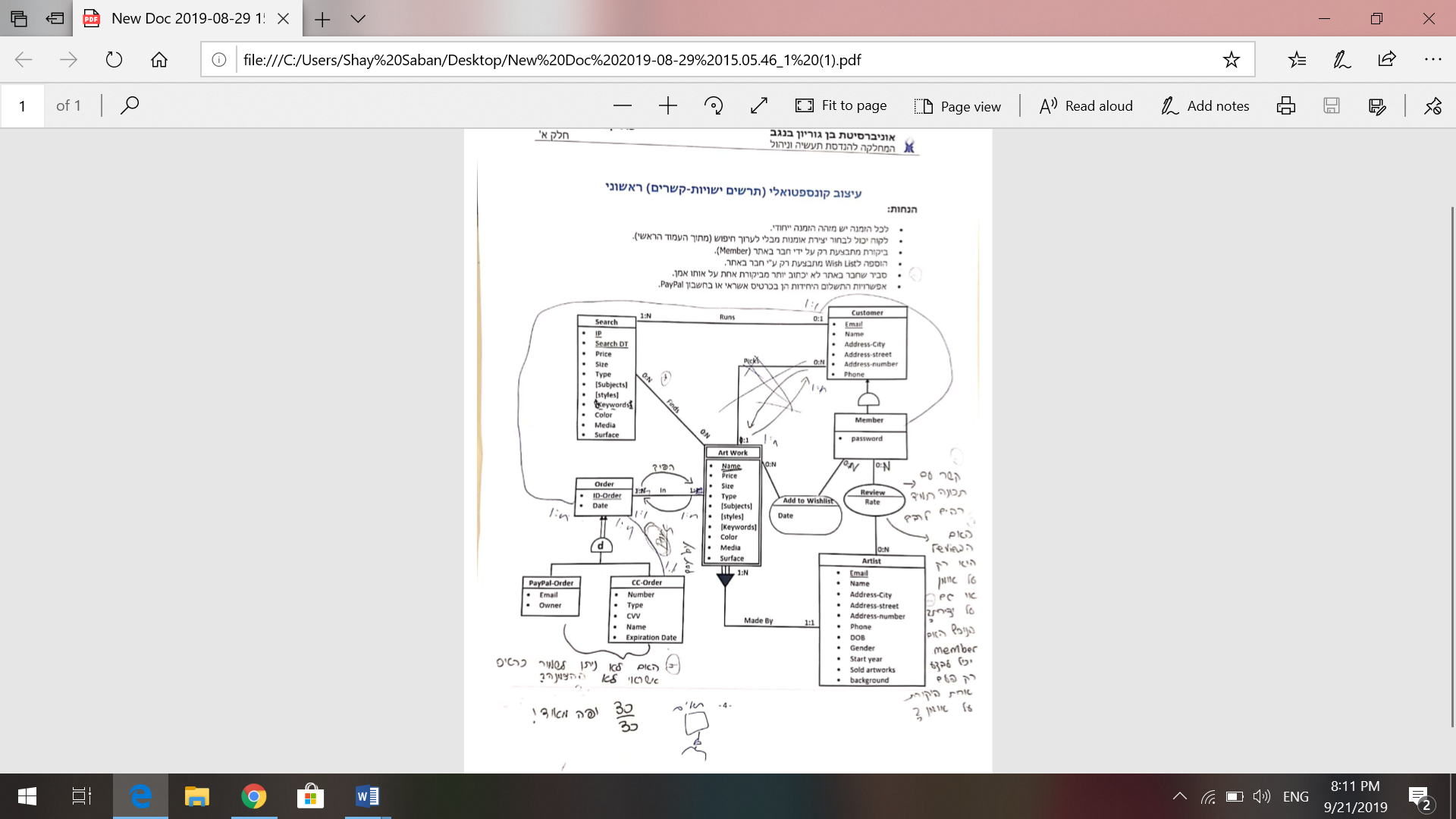
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מס' קבוצה | אתר | | | תאריך הגשה |
| 5 | https://[www.artgallery.co.uk/](http://www.artgallery.co.uk/) | | | 22/09/2019 |
| חברי הצוות - מספרי ת.ז | | | | |
|  | |  |  | |

## פרויקט בסיסי נתונים – חלק ג'

**ERD ישן:**



## מבנה טבלאי ישן:

## 

## ERD מתוקן:

**הנחות:**

* לכל הזמנה יש מזהה הזמנה ייחודי
* לקוח (Customer) מוגדר כמי שנכנס לאתר, גם אם לא הכניס את פרטיו, וכל לקוח יכול לבצע רכישה (לא רק חבר (Member)).
* ביקורת מתבצעת רק ע"י חבר באתר (Member), וביקורת יכולה להתבצע ע"י אותו חבר מספר פעמים- נעשה תיקון בקשר Review שהוחלף בישות חלשה על מנת שתוכל להיות חזרתיות וחבר יוכל לבצע ביקורת מספר פעמים.
* הוספה לWishList- מתבצעת רק ע"י חבר באתר.
* אפשרות התשלום היחידה היא בכרטיס אשראי.
* אין אפשרות להכניס פרטי תשלום באתר שלא תחת אופציית "הזמנה".

**התיקונים שבוצעו במודל ה-ERD:**

* תיקון בPayment- השדה Type היה מיותר
* תיקון בReview- שינוי לישות חלשה של Member במקום קשר כדי שניתן יהיה לבצע חזרתיות (חבר יכול לבצע כמה פעמים ביקורת לאמן, לפי ההנחה)
* הוספנו את הקשר Gets בין Artist לReview, וכן גם טבלה של הקשר כי הקשר הוא רבים לרבים
* הפכנו את Color לשדה מרובה ערכים- מימשנו טבלת lookup
* תיקון בARTISTS- הורדנו את Sold Artworks כי ניתן לראות בטבלת Orders אילו יצירות של האומן נרכשו. כלומר, שדה שניתן להוציא מהנתונים הקיימים.

**מודל נתונים טבלאי מתוקן:**

* **Customers** (Email, Name, Address\_City, Address\_Street ,Address\_Number ,Phone)
* **Members** ( Email(Customers) ,Password)
* **Searches** (IP, Search DT, Price , Size, Runs(Customers))
* **Search\_Styles** ({IP, search DT}(Searches), Styles)
* **Search\_Colors (**({IP, search DT}(Searches), Colors)
* **Artworks** (Email(Artists),Name , Price, Size, Colors, IN(Orders))
* **Artworks\_Styles** ({Email, Name}(Artworks), Style)
* **Artwork\_Colors** ({Email, Name}(Artworks), Color)
* **Finds**({IP, Search\_DT}(Searches),{Name, Email}(Artworks))
* **Artists** (Email ,Name ,Address\_City, Address\_Street, Address\_Number , Phone, DOB, Gender, Start year, Background)
* **Gets** ({Email,DT}(Member), Email(Artists))
* **Reviews** (Email(Mebmers), DT, Rate)
* **Add to Wishlists** ({Name,Artist\_Email}(Artworks), Member\_Email(Members), Date)
* **Orders** (ID-Order ,Date ,Makes(Customers), Pay by(Payments))
* **Payments** (Number, CVV, Name, Expiration date)

## מטלה 1 (25%) – שאילתות

**שתי שאילתות SELECT ללא קינון**

שאילתה 1:

מצא את שלוש האומנים שדרוגם הוא הנמוך ביותר בשלוש השנים האחרונות וכמות הפעמים שרכשו מכל אמן.

הצג את שם האומן, דירוגו הממוצע וכמות הפעמים שרכשו ממנו.

select top 3 [Artist Name]=A.Name, averegeRank=avg (rate), numOfOrders=Count(O.ID\_Order)

from Gets as G join Artists as A on A.Artist\_Email=G.Artist\_Email

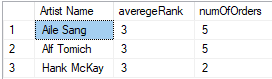
join Artworks as AW on A.Artist\_Email=AW.Artist\_Email

join Orders as O on O.ID\_Order= AW.ID\_Order

Where YEAR(O.[Date\_Order]) > year(getdate()) - 3

group by A.Artist\_Email, A.Name

order by averegeRank



מטרת השאילתה היא למצוא את האמנים שקיבלו את הדירוג הנמוך ביותר. באמצעות שאילתה זו נוכל לדעת אילו אמנים לשווק פחות בעמוד הראשי ומה הרווחיות אשר הם מספקים לאתר.

שאילתה 2:

הצג את שמות היצירות השונות שהוכנסו לwishlist ולאחר מכן נרכשו, אשר מחירן גדול מ2000 דולר וסדר אותן לפי סדר יורד של המחיר.

select [ArtWork Name]=AW.Name\_artwork, AW.Price

from Artworks as AW join WishList as W on AW.Name\_artwork=W.Name\_artwork and AW.Artist\_Email=W.Artist\_Email

join Orders as O on AW.ID\_Order=O.ID\_Order

where AW.Price>2000

Order by AW.Price desc



מטרת השאילתה היא לדעת כמה מהלקוחות אכן מבצעים את הרכישה לאחר שהכניסו את היצירות שאהבו לרשימת המשאלות שלהם. 2000$ הוא מחיר גבוה יחסית ורצינו שהבדיקה תהיה אמינה כאשר המחיר פחות סביר לקונה. על כן גם סידרנו את המחירים לפי סדר יורד, כדי שנוכל לבדוק אם קיימות הרבה יצירות בעלות מחיר גבוה יותר בהן התעניינו הלקוחות ולבסוף קנו אותן. (ישנן מספר יצירות בעלות אותו שם, אך האומן שונה, לכן יש כפילות בשמות היצירות, אך לבסוף כל יצירה נקנית פעם אחת בלבד!)

# שתי שאילתות SELECT מקוננות

שאילתה 1:

מצא את ממוצע ההזמנות ללקוח ואת הלקוחות שהזמינו מעל הממוצע. הצג את רשימת הלקוחות (שם ואימייל) אשר מספר ההזמנות שלהם גדול מהממוצע ואת מספר ההזמנות שביצעו.

select C.Customers\_Email, C.Name, Num\_Of\_Orders=count(O.ID\_Order)

from Customers as C join Orders as O on C.Customers\_Email= O.makes

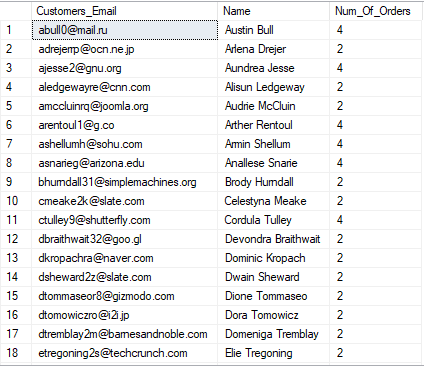
group by C.Customers\_Email, C.Name

having count(O.ID\_Order)> (select avg(Num\_Of\_Orders)

from (select C.Customers\_Email, C.Name, Num\_Of\_Orders=count(O.ID\_Order)

from Customers as C join Orders as O on C.Customers\_Email=O.makes

group by C.Customers\_Email, C.Name) as avgCustomer)



מטרת השאילתה: לאתר לקוחות רווחיים עבור האתר, לשלוח להם מיילים עם הצעות ליצירות חדשות שהועלו ובכך לעודד רכישות נוספות בעתיד.

שאילתה 2:

מצא עבור כל עיר את סך עלות ההזמנות שנעשו בה בשנה האחרונה. הצג את רשימת הערים שנעשו מהן רכישות בשנה האחרונה ואת סך עלות ההזמנות עבור כל אחת מהערים.

select City=T1.[Address-City], Profit=sum(T1.[Order\_ Sum])

from

(select C.[Address-City], O.Date\_Order, O.ID\_Order,[Order\_ Sum]=sum(AW.Price)

from Artworks as AW join Orders as O on O.ID\_Order=AW.ID\_Order join Customers as C on C.Customers\_Email=O.makes

group by O.ID\_Order, C.[Address-City], O.Date\_Order) as T1

where Year(T1.Date\_Order)>Year(Getdate())-1

group by T1.[Address-City]

order by 2 DESC



מטרת השאילתה היא להבין באילו ערים הרווחים הם גבוהים ביותר ובאילו ערים פחות. בערים שפחות, יש לעבוד על אופן שיווק האתר, לתת תמריצי מכירה כמו קופונים, הנחות, פרסום במדיה חברתית וכו' ,ובמדינות שהרווחים בהן גבוהים יותר יש לעקוב ולראות שהמצב נשמר לאורך זמן.

# שתי שאילתות מקוננות תוך שימוש במרכיבים נוספים

# שאילתה 1:

# השאילתה משתמשת בupdate, ומוסיפה עמודה בשם "isActive" לטבלת הלקוחות. השאילתה מעדכנת בעמודה "true" כלומר 1, אם הלקוח ביצע קנייה בשנה האחרונה, ו-"false" אחרת. המטרה היא לראות את מספר הלקוחות הפעילים אשר מבצעים הזמנות לפחות אחת לשנה.

ALTER TABLE Customers ADD isActive bit

update Customers

set isActive =1

where Customers.Customers\_Email in (select Orders.makes

from Orders

where YEAR(Orders.Date\_Order)<=YEAR(getdate())+1)

update Customers

set isActive =0

where Customers.Customers\_Email not in (select Orders.makes

from Orders

## where YEAR(Orders.Date\_Order)>YEAR(getdate())-1)



# שאילתה 2:

השאילתה משתמשת ב**intersect**, מצא את כל שמות הלקוחות, מספר הטלפון שלהם והמייל אשר רכשו בשנה האחרונה מוצר בסגנון "Digital art" . לכבוד פתיחת מוזיאון חדש, המתמקד ביצירות דיגיטליות חדשניות, שמתקיימת בעוד כחודש, נוצר שיתוף פעולה בין האתר למוזיאון בו מוגרלים כרטיסים חינם ללקוחות אשר רכשו יצירה בסגנון Digital art"" מהאתר בשנה האחרונה.

select X.Customers\_Email, X.Name, X.Phone, X.Date\_Order

from (select [Style]=AWS.Style,O.ID\_Order, C.Customers\_Email, C.Name, C.Phone, O.Date\_Order

from [Artwork-Styles] as AWS join Artworks as AW on AWS.Artist\_Email= AW.Artist\_Email join Orders as O on AW.ID\_Order=O.ID\_Order join

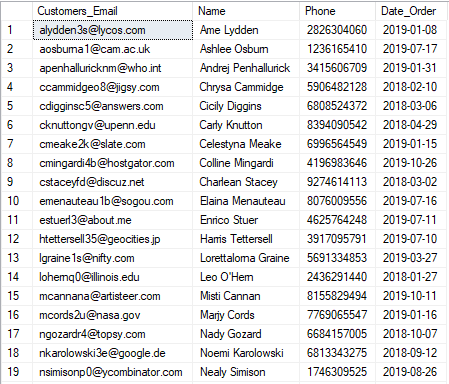
Customers as C on C.Customers\_Email=O.makes) as X

where X.Style ='Digital art' AND Year(X.Date\_Order)>=YEAR(getdate())-1

intersect

select C.Customers\_Email, C.Name, C.Phone, O.Date\_Order

from Customers as C join Orders as O on C.Customers\_Email=O.makes



## מטלה 2 – העשרת בסיס הנתונים

# View

ה-VIEW מרכז בתוכו את כל הפרטים של הלקוח וההזמנה שלו במקום אחד לטובת מחלקת המשלוחים. הוא מכיל את כלל פרטי ההזמנה הרלוונטיים לשליח כמו שם הלקוח, טלפון הלקוח, הכתובת אליה הזמין הלקוח את היצירה, מזהה ההזמנה והשם של משלם ההזמנה.

ה-VIEW מחסיר פרטים של הלקוחות כמו מספר כרטיס אשראי וCVV, על מנת לשמור על פרטיות הלקוח. בנוסף מחסיר גם פרטים מיותרים כמו Email, Gender, Order\_Date.

Create view dbo.Details\_For\_Delivery as

select CU.Name as [Name], CU.Phone as [Phone Number], O.ID\_Order as [Order ID], CU.Country as [Delivery Country], CU.[Address-City] as [Delivery City], CU.[Address-Street] as [Delivery Street], CU.[Address-Number] as [Delivery St.Number], P.Name as [Payed By]

from Customers as CU join Orders as O on CU.Customers\_Email=O.makes

join Payment as P on P.Number=O.PayBy

group by CU.Name , CU.Phone, O.ID\_Order, CU.Country, CU.[Address-City], CU.[Address-Street], CU.[Address-Number], P.Name

GO

דוגמה לשימוש ב-VIEW:

select \* from dbo.Details\_For\_Delivery



# פונקציות (Functions)

פונקציה שמחזירה טבלה:

האתר נוהג לשלוח מתנת יום הולדת לאמנים הרשומים אליו, בחודש הולדתם. כדי שמנהלי האתר יוכלו לעקוב אחרי האמנים להם צריך לשלוח את המתנה יצרנו את הפונקציה BirthdayArtist, אליה מכניסים את החודש הרצוי והיא מחזירה את טבלת האמנים להם יש יום הולדת באותו חודש. הטבלה תכיל את שם האמן, כתובת האימייל שלו ותאריך הלידה.

CREATE FUNCTION BirthdayArtists (@CurrMonth int)

RETURNS TABLE

AS RETURN

SELECT A.Name AS [Name], A.Artist\_Email AS [Email], A.DOB as [Date Of Birth]

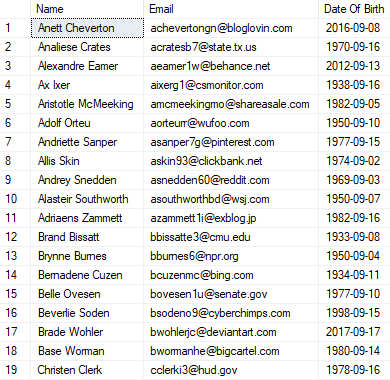
FROM Artists as A

WHERE Month(A.DOB)=@CurrMonth

דוגמה לשימוש בפונקציה:

select \*

from BirthdayArtists(09)



פונקציה סקלארית:

פונקציה שמקבלת שנה ושם של מדינה ומחזירה את סכום ההזמנות. נשתמש בפונקציה על מנת להבין מה סכום ההזמנות שנעשות במדינות מסוימות והאם רווחי לחברה להשאיר את המדינה בתור קונה פוטנציאלית, או שמא החברה מרוויחה מאותה מדינה פחות ממה שהיא משלמת על שכר לשליחים למשל.

CREATE FUNCTION SumOrdersInCountry (@year int, @Country Varchar(20))

RETURNS Int

AS BEGIN

declare @SumOfOrdersInCountry int

SELECT @SumOfOrdersInCountry=sum(A.Price)

FROM Orders as O join Customers as C on O.makes=C.Customers\_Email join Artworks as A on A.ID\_Order=O.ID\_Order

WHERE @Country= C.Country and Year(O.Date\_Order)=@year

RETURN @SumOfOrdersInCountry

END

דוגמה לשימוש בפונקציה:

select [Sum Of Sales]=[bgu-users\sabans].[SumOrdersInCountry] (2019,'Brazil')



# Trigger פשוט

פעולה אשר מעדכנת עבור כל אמן את כמות היצירות שהעלה לאתר. הטריגר- בעת כל הכנסת יצירה חדשה של האמן- מעדכנת את השדה בטבלת Artists. הפעולה העסקית שעומדת מאחורי הטריגר, היא לדעת כמה יצירות יש לאותו אמן בכל רגע נתון ולפיכך נוכל לקבל את היחס בין כמות היצירות שהעלה לכמות היצירות שמכר.

alter table Artists

add [Number Of ArtWorks] int default 0

update Artists

set [Number Of ArtWorks]=0

alter table Artists

DROP COLUMN [Number Of ArtWorks]

CREATE TRIGGER Update\_NumArtworks1

ON Artworks

FOR INSERT, UPDATE

AS UPDATE Artists

SET [Number Of ArtWorks]= [Number Of ArtWorks]+(SELECT Count(\*) FROM INSERTED

Where dbo.Artists.Artist\_Email = INSERTED.Artist\_Email)

# פרוצדורה שמורה ((Stored Procedure פשוטה

יצרנו פרוצדורה שמורה שתאפשר לחבר באתר לעדכן את הסיסמה האישית שלו.

ההצדקה לשימוש בפרוצדורה השמורה הינה לאפשר לחבר באתר לשנות את הסיסמה שלו, אך לא לאפשר לו גישה לשאר הנתונים בבסיס הנתונים כמו אימייל, שם וכו'.

CREATE PROCEDURE updatePassword

@Email Varchar(20),@Password Varchar(20)

AS

UPDATE dbo.Members

set Password=@Password

Where Members.Customers\_Email=@Email

Select \*

From Members

Where Customers\_Email=@Email

להלן נתוני הלקוחה Arlena Drejer לפני השימוש בפרוצדורה השמורה:



דוגמה לשימוש בפרוצדורה השמורה:

exec updatePassword 'adrejerrp@ocn.ne.jp', '442HP13'



## מטלה 3 (20%) – כלים להצגת נתונים

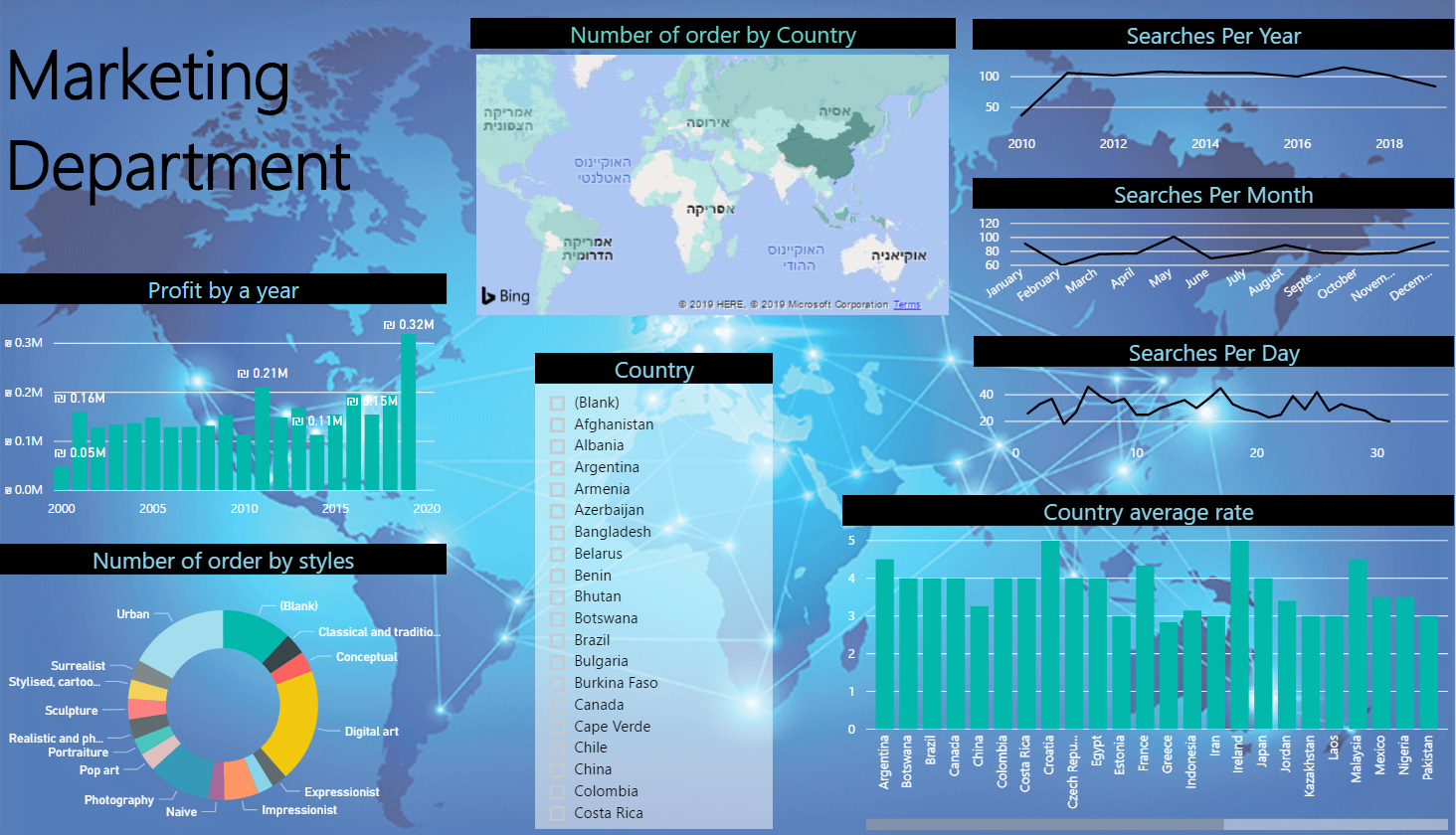
**דו"ח עסקי -**

הדו"ח הבא הינו דו"ח המיועד למחלקת השיווק של האתר. על מנת לאתר תנועת גולשים ברשת, ועל פי נתונים אלו לנתח את אסטרטגיית השיווק של האתר- באיזה שעות לפרסם, באיזה מדינות קיים ביקוש גדול יותר ליצירות ובאיזה מדינות פחות (ומתוך כך באיזה מדינות יש להגביר פרסום), וכדומה.

הדו"ח מציג את הנתונים הבאים:

חיפושים- מידע על חיפושים נותן מידע חשוב ורלוונטי ביותר בכדי להבין אם יש התקדמות משנה לשנה בכמות הכניסות והחיפושים באתר, כדי לתת ביקורת פנימית למחלקת השיווק ,כמו כן ישנם חודשים בהם הפעילות באתר גבוהה יותר, וחשוב לא פחות הימים והחודשים בהם תרבות הצריכה קטנה יותר מכיוון שבימים אלה יש להגביר פעילות שיווקית.

* חיפוש באתר על פי יום
* חיפוש באתר על פי חודש
* חיפוש באתר על פי שנה
* מספר הזמנות לפי מדינה -מידע זה חשוב ורלוונטי למחלקת השיווק מכיוון שהוא מראה בדיוק היכן שיווק האתר פועל בצורה טובה והיכן פחות ,כאשר במדינות בהן מספר ההזמנות נמוך יש לתגבר פעילות שיווקית. הנתון מוצג וויזואלית על ידי מפה ,ככל שהצבעים יותר כהים כך מספר ההזמנות מהמדינה גדול יותר.
* רווחים לפי שנה -רווחים מראים לנו המון מידע שיכול לעזור למחלקת השיווק מכיוון שישנן מדינות עניות ומדינות עשירות יותר, מדינות עם תרבות צריכה אינטרנטית גבוהה יותר וכאלה עם פחות. מידע זה רלוונטי וחשוב ביותר.
* מספר ההזמנות לפי סגנון – מידע זה חשוב מאוד למחלקת שיווק מכיוון שישנם סגנונות פופולריים יותר ופחות. מידע זה יכול להשפיע אילו יצירות נרצה לדוגמא לשים בעמוד הראשי ואילו יצירות אינן נגישות מספיק או שפחות מתעניינים בהן.

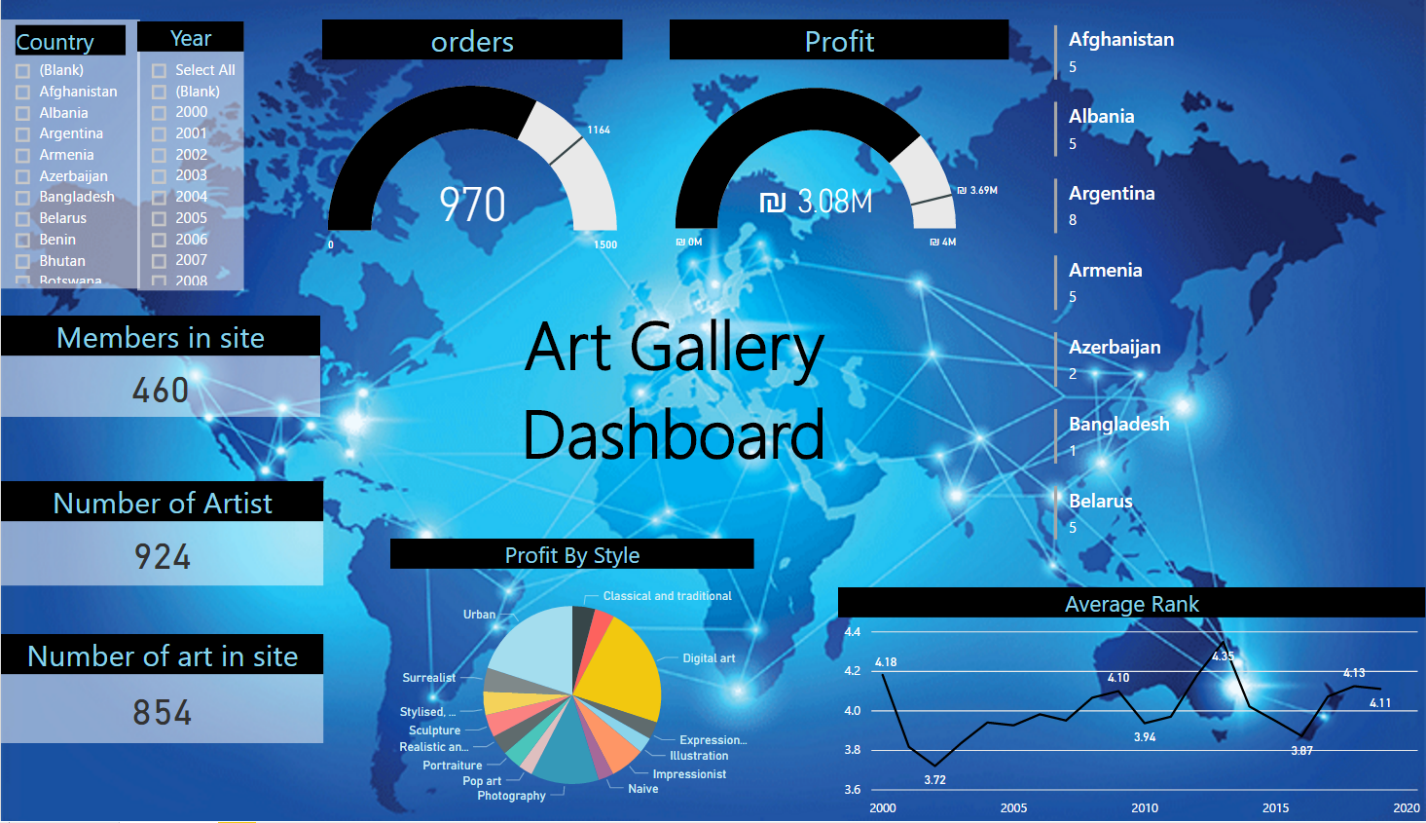
****

**לוח מחוונים-** הדו"ח מיועד לדרג הניהולי הבכיר של הארגון, ונועד לתמוך בקבלת ההחלטות שלהם על בסיס נתונים מתעדכנים מעת לעת. הדו"ח נוצר בפלטפורמת Power BI. עיקר הדו"ח עוסק בניתוח גלובאלי של נתוני מכירות ברחבי העולם, כמו גם נתונים אודות מוצרים ולקוחות.

הנושאים העסקיים אותם מציג הדו"ח:

* ההכנסה הכוללת של הארגון- חיפוש זה יבוצע באמצעות בחירת שנה מבוקשת ומדינה מבוקשת. בבחירת שנה ומדינה יוצגו ההכנסות מהמדינה בשנה הנבחרת. בבחירת שנה בלבד או מדינה בלבד יוצגו ההכנסות עבור השנה הנבחרת מכלל המדינות או עבור המדינה הנבחרת בכל השנים. ההכנסות מוצגות אל מול היעד שהוגדר להיות 4,000,000 ₪.
* כמות ההזמנות במערכת אל מול היעד- נתון מתעדכן על פי השנה הנבחרת והמדינה הנבחרת שהמטרה היא לגדול ב20 אחוזים כל שנה בכמות ההזמנות.
* ממוצע הדירוג לפי מדינות- פיצ'ר זה מתעדכן אל מול בחירת השנה והמדינה המבוקשת, ומציג את הנתונים הרלוונטיים.
* רווח לפי סגנון – מידע חשוב ורלוונטי למנהלי החברה ,בארגון כמו שלנו שלכל יצירה יש סגנון, מנהלי החברה יצטרכו לדעת מהם הסגנונות המובילים והאם כדאי להתמקד בהם.

נתונים אלו מהווים מידע חשוב למנהלי הארגון, כלים אלה יעזרו למנהלים לקבל החלטות חשובות בנוגע לעתיד הארגון וכיוונו. מידע זה שהצגנו יכול, לאחר הצלבה של נתונים, בהחלט לעזור למנהלים במידע זה. ככל שהמידע אמין יותר ורלוונטי, ראשי הארגון יקבלו החלטות קלות וטובות יותר.



על מנת לממש את הדו"ח מימשנו את הVIEWS המפורטים להלן:

---Increase Orders by 20 percent---

create view [digital dashbored Orders] as

select [target]=COUNT( o.ID\_Order)\*1.2,O.ID\_Order

from Orders as O

group by O.ID\_Order

---Increase income by 20 percent---

create view [digital dashbored profits] as

select [Target] = (Sum(AW.Price))\*1.2

from Orders as O join Artworks as AW on O.ID\_Order=AW.ID\_Order

group by Year(O.Date\_Order)

## מטלה 4 – יישומים מתקדמים

## כלי מורכב (Stored Procedure או Trigger)

**4.1 טריגר מורכב עם Cursor:**

הטריגר משמש את מחלקת הביטחון של החברה. מטרתו היא לזהות האם בוצעו 2 רכישות או יותר בעזרת אותו כרטיס אשראי באותו יום, או לחילופין רכישה בסכום של 10000 דולר או יותר. מצב זה יכול להעיד על גניבת כרטיס אשראי. יצרנו טבלת dbo.Securitiesאשר תכיל את פרטי ההזמנות שזוהו ככאלה שעומדות בקריטריונים הנ"ל.

CREATE TABLE dbo.Securities (

Card\_Nuber VARCHAR(16),

ID\_Order int,

Date date)

CREATE TRIGGER SecurityCheck

ON Orders

after INSERT

AS

Declare @ID\_Order Varchar(20)

Declare @Date Date

DECLARE @Card\_Number Real

DECLARE C\_CURSOR2 CURSOR FOR

SELECT [PayBy],[ID\_Order],[Date\_Order]

From Inserted

OPEN C\_CURSOR2

FETCH NEXT FROM C\_CURSOR2

INTO @Card\_Number,@ID\_Order,@Date

WHILE (@@FETCH\_STATUS = 0)

BEGIN

IF (select count(Orders.PayBy)

from Orders

Where Orders.PayBy=@Card\_Number)>2 And

(Select count(Orders.Date\_Order)

from Orders

where Orders.Date\_Order=@Date and Orders.PayBy=@Card\_Number

group by Orders.Date\_Order)>2

or (SELECT [Total Price Per Order] = sum (Artworks.Price)

FROM Orders join Artworks on Orders.ID\_Order=Artworks.ID\_Order

where Orders.ID\_Order=@ID\_Order ) >10000

Begin

insert into dbo.Securities

select inserted.PayBy, inserted.ID\_Order, inserted.Date\_Order

From inserted

where inserted.PayBy=@Card\_Number And inserted.ID\_Order=@ID\_Order and inserted.Date\_Order=@Date

END

Else

PRINT 'OK'

FETCH NEXT FROM C\_CURSOR2

INTO @Card\_Number,@ID\_Order,@Date

END

---------------------

CLOSE C\_CURSOR2;

DROP TABLE [Securities]

שורה שהפעילה את הטריגר:

Insert Into [Orders]

Values

('11-111-1112', '2017-04-16', 'acallangm@over-blog.com', '4962911945')

## דו"ח המושתת על שאילתה מקוננת מורכבת

## --שאילתה אשר מחזירה רק את הלקוחות אשר ביצעו הזמנה, וכתבו ביקורת אחת לפחות--

CREATE VIEW customersList

AS

select C.Customers\_Email

from Customers as C

intersect

select R.Customers\_Email

from Reviews as R

group by R.Customers\_Email

intersect

select O.makes

from orders O

group by O.makes

השאילתה אשר מחזירה את כמות הפעמים שכל לקוח חיפש סגנון----

create view Look\_For\_Style

as

select [Customer\_Email]=S.Runs, [IP] = SS.IP, SS.Style, NumOfSearches = count (SS.Style)

from Searches as S join [Search-Styles] as SS on S.IP=SS.IP

group by SS.IP,SS.Style, S.Runs

--שאילתה אשר מחזירה את הסגנון שכל לקוח חיפש הכי הרבה--

create view Most\_Searchable\_Style

as

select [Customer\_Email]=S.Runs, [IP] = S.IP, favouriteStyle= (select L.Style

from Look\_For\_Style as L

where L.Customer\_Email = S.Runs)

from Searches as S

group by S.Runs, S.IP

having (select max(L.NumOfSearches)

from Look\_For\_Style as L

where L.Customer\_Email = S.Runs) >0

שאילתה אשר מחשבת את רמת שביעות הרצון של כל הלקוחות לפי ממוצע הביקורות אשר רשם באתר----

create Function dbo.calc\_satisfaction (@customerEmail varchar(20))

returns varchar (20)

as

begin

declare @satisfaction varchar(20)

if (select avgReviews= avg(R.Rate)

from Reviews as R

where R.Customers\_Email = @customerEmail

group by R.Customers\_Email) > 4.1

set @satisfaction = 'satisfied'

else if (select avgReviews= avg(R.Rate)

from Reviews as R

where R.Customers\_Email =@customerEmail

group by R.Customers\_Email) < 2.1

set @satisfaction = 'unhappy'

else

set @satisfaction = 'neutral'

return @satisfaction

end

שאילתה מקוננת אשר מייצרת את הדוח----

הדוח מציג עבור כל לקוח את רמת שביעות הרצון שלו מהמוצרים, את סך הרכישות שביצע, סך החיפושים, וסגנון מועדף--

select C.Customers\_Email, C.Name, satisfaction = dbo.calc\_satisfaction(C.Customers\_Email),

numOfOrdersMade = (select count(\*)

from Orders O

where O.makes= C.Customers\_Email

group by O.makes),

totalAmountOfPurchases = ( SELECT sum(AW.Price)

from Artworks as AW join orders O on AW.ID\_Order=O.ID\_Order

where C.Customers\_Email = O.makes

group by O.makes),

numOfSearches= (select NumOfSearches

from Look\_For\_Style L

where L.Customer\_Email = C.Customers\_Email),

favourite\_Category = (select favouriteStyle

from Most\_Searchable\_Style as MSS

where MSS.Customer\_Email = C.Customers\_Email)

from customers C

group by C.Customers\_Email, C.Name

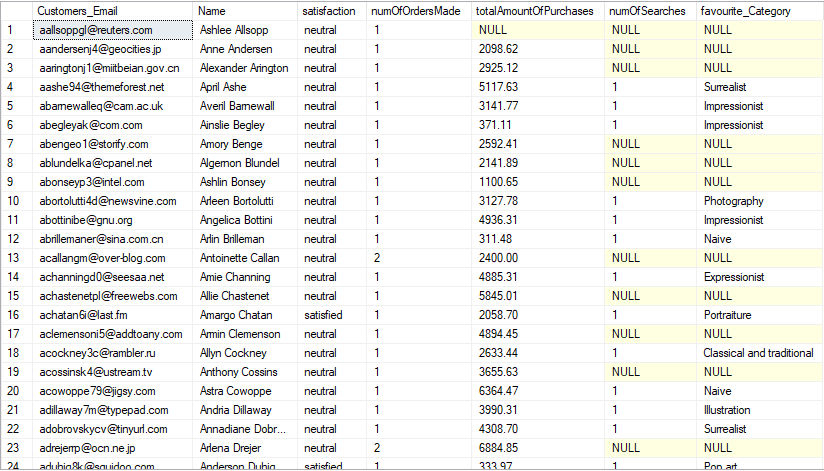
having (select count(\*)

from Orders O

where O.makes=C.Customers\_Email

group by O.makes) >0

הדו"ח שהתקבל:



## שילוב מערכתי של מספר כלים

השאילתה שעשינו בודקת עבור הוספת שורה חדשה ל-Payments האם כרטיס האשראי הוא פג תוקף.

במידה וכן(כרטיס האשראי הוא פג תוקף: השאילתה לא מוסיפה את הרשומה. במידה וכרטיס האשראי הינו בתוקף(תקין), הרשומה מתווספת.

הפונקציה מחזירה 1 אם כרטיס האשראי בתוקף, אחרת מחזירה 0.

CREATE function dbo.IsCreditVailed (@ExpirationDate DATE)

returns int as

begin

if YEAR(@ExpirationDate)< YEAR(getdate()) or

(YEAR(@ExpirationDate) =YEAR(getdate()) and MONTH(@ExpirationDate)<month(getdate()))

return 0

return 1

end

--drop function dbo.IsCreditVailed

create procedure Deleteornot @expirationDate DATE

AS DECLARE @isvailed INT

BEGIN

set @isvailed = dbo.IsCreditVailed(@expirationDate)

IF @isvailed = 0

delete from dbo.Payment where [Expiration\_Date] = @expirationDate

end

create trigger updateCreditcards on dbo.Payment

FOR insert as

DECLARE @expirationDate DATE

BEGIN

select @expirationDate = [Expiration\_Date] from inserted

execute deleteornot @expirationDate

end

insert into dbo.Payment values('098204612343064', '345', 'Yoni','1993-08-20')

insert into dbo.Payment values('198204612343064', '345', 'Shay','2023-08-20')

SELECT dbo.IsCreditVailed ('1993-08-20')

SELECT dbo.IsCreditVailed ('2023-08-20')

SELECT \* FROM dbo.Payment ORDER BY [Expiration\_Date]

# בהצלחה!