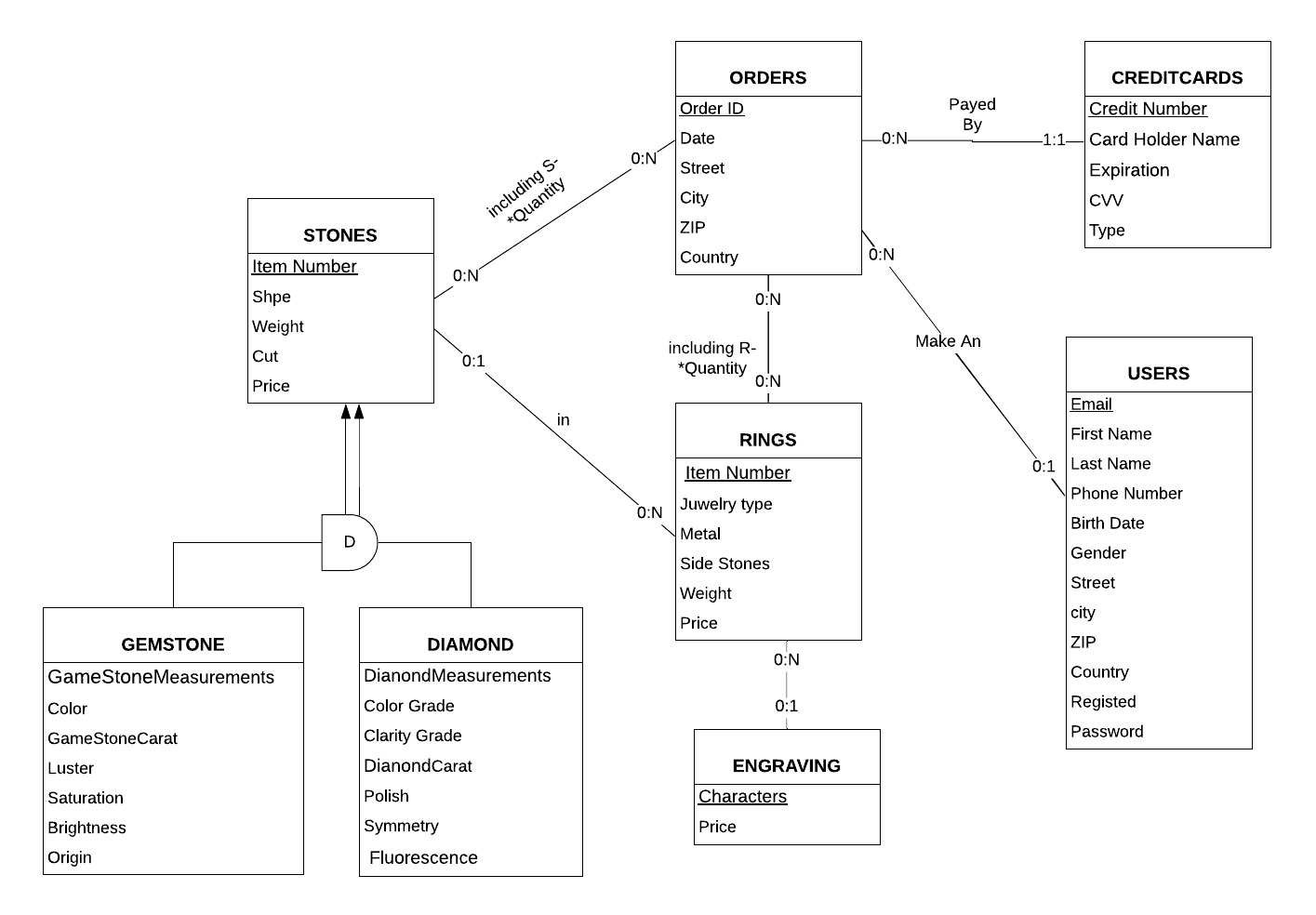
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מס' קבוצה | אתר | | | תאריך הגשה |
| 16 | [www.israel-diamonds.com](https://www.israel-diamonds.com) | | | 19/01/2020 |
| חברי הצוות - מספרי ת.ז | | | | |
| 206952863 | | 314963810 | 206087736 | |

## פרויקט בסיסי נתונים – חלק ב'

**ERD מתוקן:**



**עיקרי התיקונים:**

- הוספנו ישות חריטה.

- שינינו את הקרדינליות בין ישות הזמנה לישות טבעת, היות וכל טבעת יכולה להיות במספר

הזמנות.

- שינינו את הקרדינאליות בין ישות הזמנה לישות אבן, היות וכל אבן יכולה להיות בכמה הזמנות.

**מבנה טבלאות מתוקן:**

* **STONES** (Item Number, , Shape, Weight, Cut, Price)
* **CREDITCARDS** (Credit Number, Card Holder Name, Expiration, CVV , type)
* **USERS** (Email, First Name, Last Name, Phone Number, Birth Date, Gender, Street, City, ZIP, Country, Registed, password)
* **RINGS** (Item Number, Jewelry type, Metal, Side Stones, Weight, Price , Stone Number (STONES), Engraving(ENGRAVING))
* **ORDERS** (Order ID, Date, Street, City, ZIP, Country, Credit Number(CREDITCARDS) , Email (USERS))
* **DIAMONDS** (Item Number (STONES), DiamondMeasurements, Color Grade, Clarity Grade, DiamondCarat, Polish, Symmetry, Fluorescence)
* **GEMSTONES** (Item Number(STONES), GameStoneMeasurements, Color,GameStoneCarat, Luster, Saturation, Brightness, Origin)
* **ORDERRING** (Order ID(ORDERS) , Ring ID (RINGS) , Quantity , Size)
* **ORDERSTONE** (Order ID(ORDERS) , StoneID(STONES) , Quantity)
* **ENGRAVING** (Characters , price)

## מטלה 1 (25%) – שאילתות

# שתי שאילתות SELECT ללא קינון 5%

**שאילתה 1**

החברה מעוניינת לדעת באילו מדינות להשקיע בפרסום ולכן רוצה לבדוק את ממוצע סך הפריטים שנרכשו בהזמנה בודדת מכל מדינה ולשבץ את המדינות מהגבוה לנמוך לפי ממוצע זה בשנה החולפת . בנוסף, תרצה להציג את כמות הפריטים הכוללת שהזמינו מארצות אלו , כמות הטבעות וכמות האבנים (יהלומים ואבני חן ) כאשר ה2 האחרונות חייבות להיות גדולות מ 0 , כלומר מדינות שהזמינו בהם לפחות טבעת\יהלום אחד :

Select top 10 Country ,[Number of orders] = Count(country) , [Total products orderd] = (isnull(sum(ORDERRING.Quantity),0) + isnull(sum(Orderstone.Quantity),0)) , [Total rings orderd] = sum(Orderring.Quantity) , [Total stones orderd] = sum(Orderstone.Quantity) , AVERAGE = ((isnull(sum(ORDERRING.Quantity),0) + isnull(sum(Orderstone.Quantity),0)) / Count(country))

From ORDERS left outer join Orderring on ORDERS.[Order ID]= orderring.[Order ID] left outer join ORDERStone on ORDERS.[Order ID] = orderstone.[Order ID]

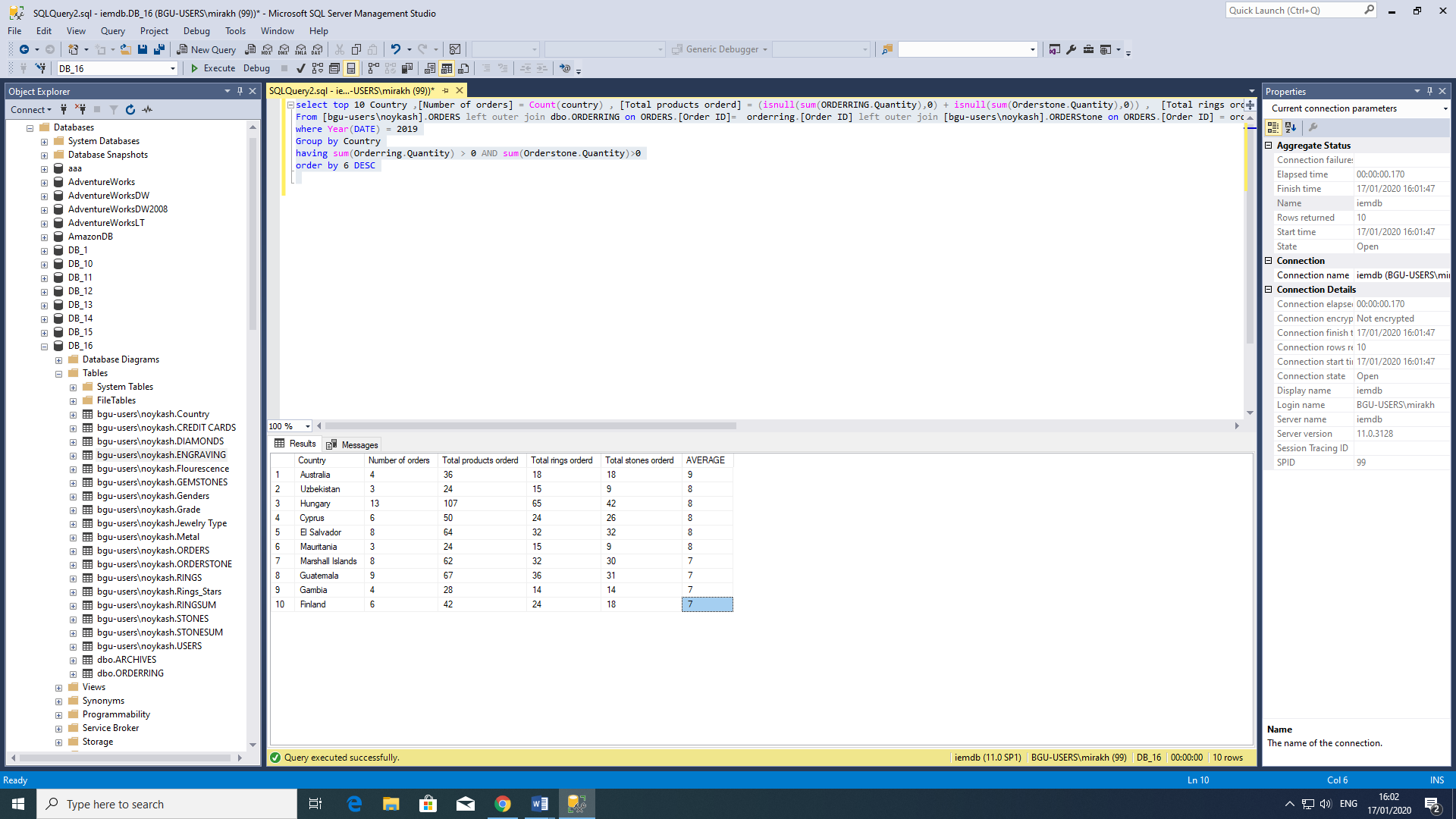
where Year(DATE) = 2019

Group by Country

having sum(Orderring.Quantity) > 0 AND sum(Orderstone.Quantity)>0

order by 6 DESC

**פלט הSQL –**



**שאילתה 2-**

החברה רוצה לעודד קניית יהלומים בקרב לקוחותיה, לכן מחלקת השיווק בחברה מעוניינת לדעת מי היו 5 הלקוחות שקנו יהלומים בסכום הגבוה ביותר בשנה החולפת וכדי שימשיכו לקנות מהאתר , יישלחו להם קופוני הנחה לרכישה הבאה:

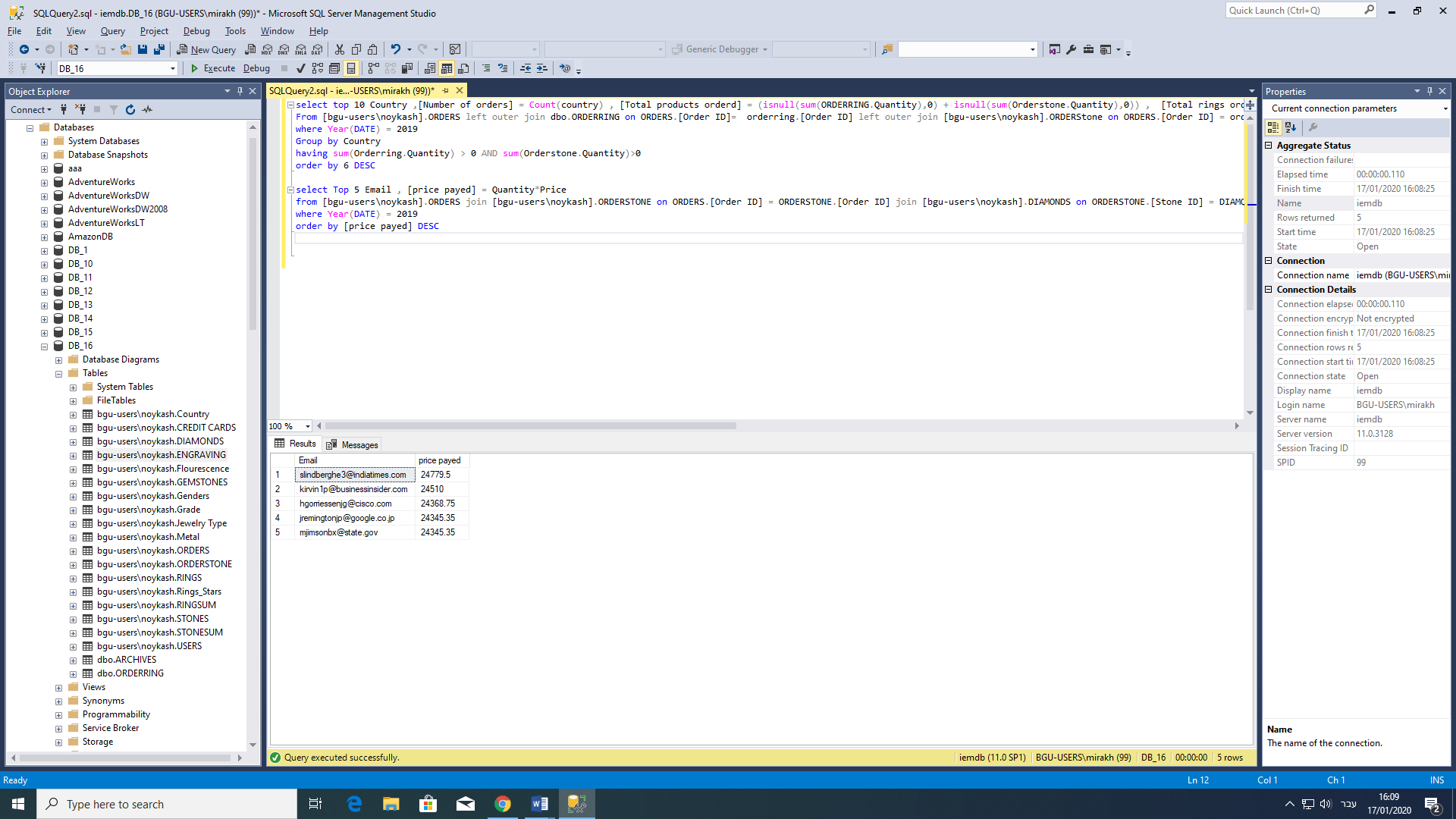
select Top 5 Email , [price payed] = Quantity\*Price

from ORDERS join ORDERSTONE on ORDERS.[Order ID] = ORDERSTONE.[Order ID] join DIAMONDS on ORDERSTONE.[Stone ID] = DIAMONDS.[Item Number] join STONES on Stones.[Item Number] = DIAMONDS.[Item Number]

where Year(DATE) = 2019

order by [price payed] DESC

**פלט הSQL –**



# שתי שאילתות SELECT מקוננות 10%

**שאילתה 1-**

**שאילתה עם קינון ב FROM שמחזיר ערך יחיד - SCALAR**

החברה רוצה לקדם קשרים עם חברת האשראי ממנה קנו הכי הרבה באתרה , לכן החליטה לבדוק מי היא חברת האשראי ולתגמל את לקוחותיה במבצעים עתידיים . כדי לא לחלק מבצעים לאלפי לקוחות ולהיגרר להפסדים, המבצעים יהיו תקפים רק במדינה ממנה הזמינו הכי הרבה באמצעות כרטיס האשראי של החברה :

select top 1 Country

from [CREDIT CARDS] join ORDERS on [CREDIT CARDS].[Credit Number] = ORDERS.[Credit Number]

where Type = (select top 1 Type

from [CREDIT CARDS] join ORDERS on [CREDIT CARDS].[Credit Number] = ORDERS.[Credit Number]

where Year(DATE) = 2019

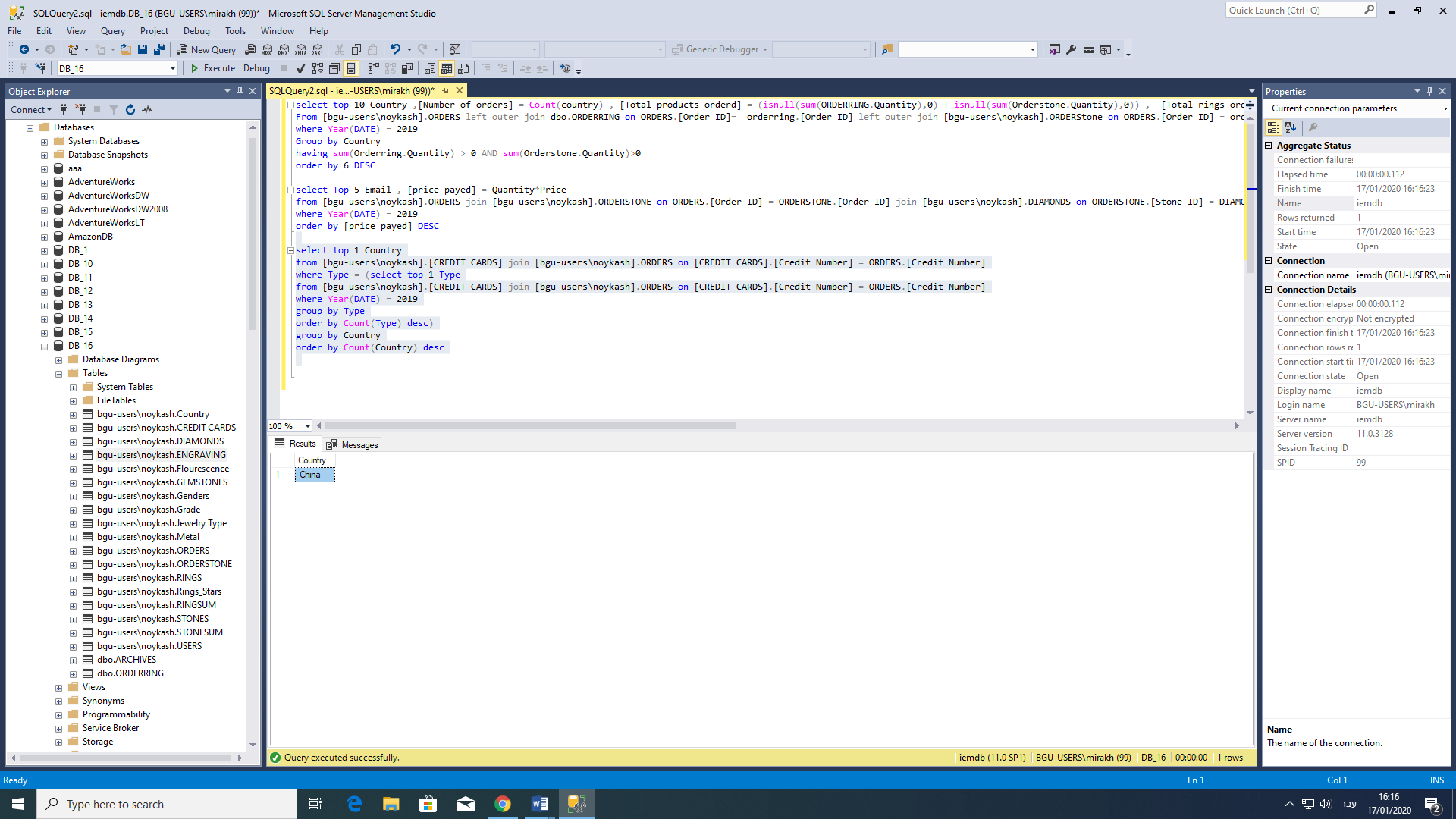
group by Type

order by Count(Type) desc)

group by Country

order by Count(Country) desc

**פלט הSQL –**



**שאילתה 2-**

**שאילתה עם קינון ב WHERE שמחזיר רשימה – LIST**

החברה מוכרת אבני חן מארצות ייצור שונות ( ORIGIN ). החברה רוצה לדעת באיזה ארצות בהם יוצרו אבני החן שמוצעות למכירה באתר שלה היא לא מכרה בכלל, כדי להפסיק לקנות מארצות אלו אבנים :

Select origin

from GEMSTONES

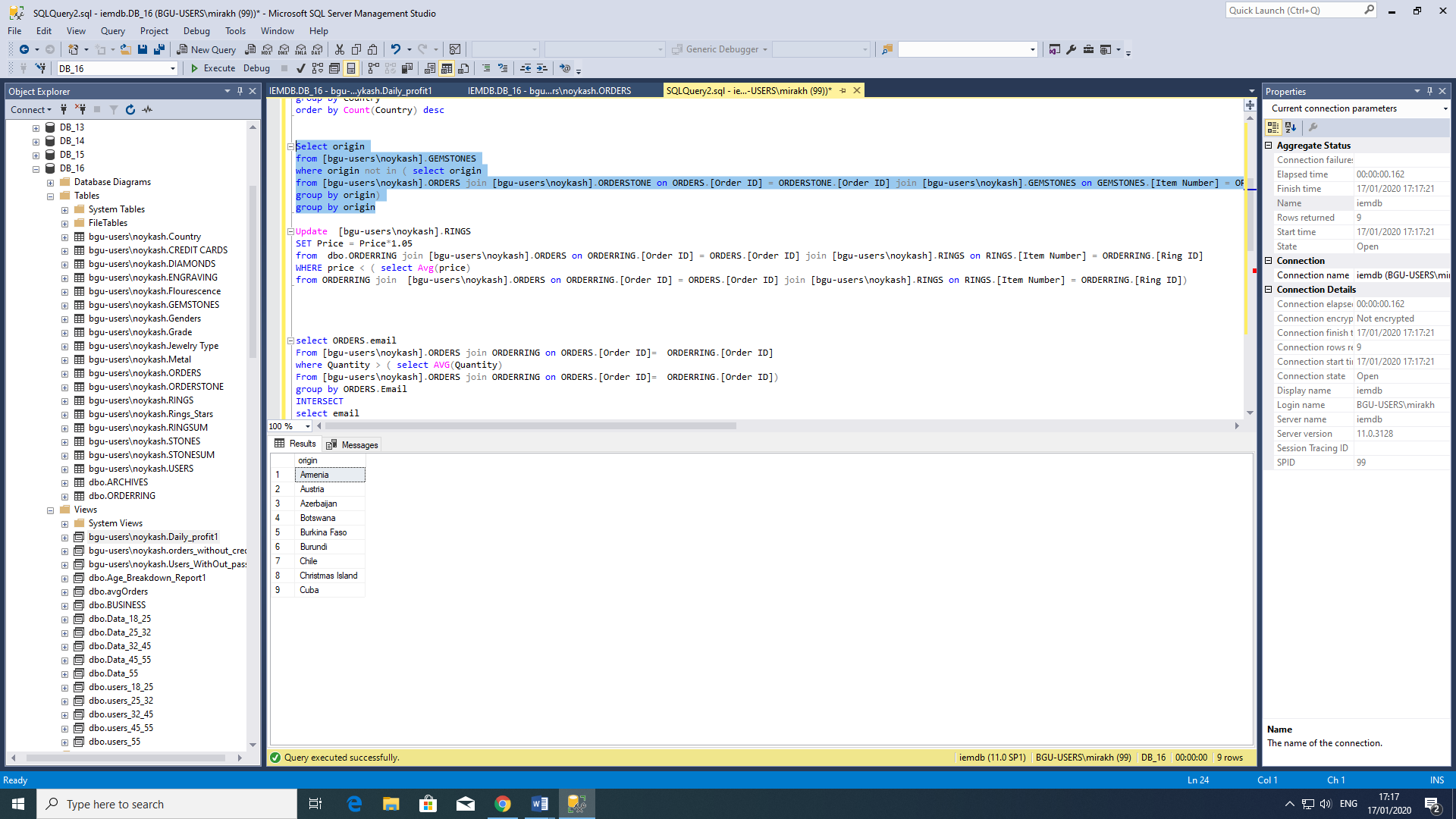
where origin not in ( select origin

from ORDERS join ORDERSTONE on ORDERS.[Order ID] = ORDERSTONE.[Order ID] join GEMSTONES on GEMSTONES.[Item Number] = ORDERSTONE.[Stone ID]

group by origin)

group by origin

**פלט הSQL –**



# שתי שאילתות מקוננות תוך שימוש במרכיבים נוספים 10%

**שאילתה 1-**

**שאילתה של UPDATE:**

החברה רוצה להגדיל את הכנסותיה ומחליטה להעלות ב **5%** את מחיר הטבעות (שנמכרו לפחות פעם אחת) ושמחירם קטן מן המחיר הממוצע מבין כל הטבעות (שנמכרו לפחות פעם אחת) :

Update RINGS

SET Price = Price\*1.05

from ORDERRING join orders on ORDERRING.[Order ID] = ORDERS.[Order ID] join RINGS on RINGS.[Item Number] = ORDERRING.[Ring ID]

WHERE price < ( select Avg(price)

from ORDERRING join ORDERS on ORDERRING.[Order ID] = ORDERS.[Order ID] join RINGS on RINGS.[Item Number] = ORDERRING.[Ring ID])

**שאילתה 2-**

**שאילתה עם INTERSECT:**

החברה רוצה לדעת מי המשתמשים אשר בכל רכישה מזמינים כמות מוצרים גבוהה מהכמות הממוצעת במטרה לעודד אותם לבצע רכישות נוספות מהאתר, לכן היא מחפשת את המשתמשים שקנו כמות טבעות בהזמנה מעל לכמות הממוצעת (להזמנה) וגם כמות אבנים בהזמנה ( יהלומים ואבני חן ) מעל לכמות הממוצעת :

select ORDERS.email

From ORDERS join ORDERRING on ORDERS.[Order ID]= ORDERRING.[Order ID]

where Quantity > ( select AVG(Quantity)

From ORDERS join ORDERRING on ORDERS.[Order ID]= ORDERRING.[Order ID])

group by ORDERS.Email

INTERSECT

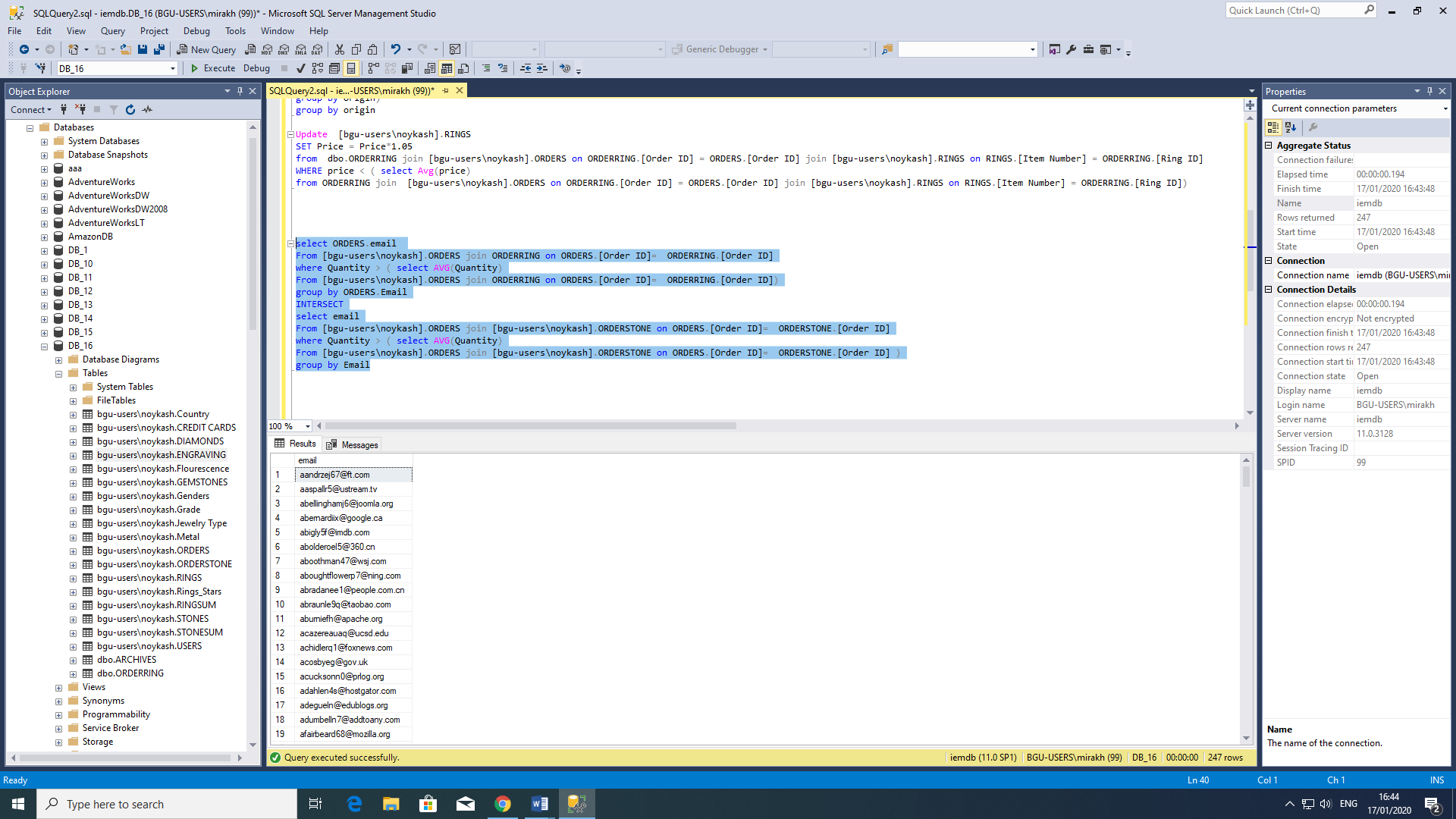
select email

From ORDERS join ORDERSTONE on ORDERS.[Order ID]= ORDERSTONE.[Order ID]

where Quantity > ( select AVG(Quantity)

From ORDERS join ORDERSTONE on ORDERS.[Order ID]= ORDERSTONE.[Order ID] )

group by Email

**דוגמא לפלט הSQL – קיבלנו 247 רשומות.**

## מטלה 2 (25%) – העשרת בסיס הנתונים

# View 5%

**VIEW 1:**

יצרנו VIEW המציג את סכום ההכנסות היומי של החברה .המטרה העיסקית שלנו בפעולה זו היא מעקב אחר ההכנסות היומיות של החברה. החברה מעוניינת להישאר במעקב אחר הכנסותיה מהאתר וכדי שבכל רגע נתון יוכלו להיכנס לבדוק את מצב ההכנסה יש צורך תמידי בעידכון הנתונים בהתאם להזמנות שנוספות.

create view Daily\_profit1

as

select profit=(sum(R.Price) + SUM(S.Price))

FROM ORDERS AS o JOIN ORDERRING AS orR on O.[Order ID] = orR.[Order ID]

JOIN ORDERSTONE as orS on O.[Order ID] = orS.[Order ID]

JOIN RINGS as R on R.[Item Number] = orR.[Ring ID]

JOIN ENGRAVING as E on R.Engraving = e.Characters

JOIN STONES as S on S.[Item Number] = orS.[Stone ID]

where O.Date = GETDATE()

**VIEW 2:**

יצרנו VIEW המציג את טבלת המשתמשים תוך הסתרת פרטים רגישים, לדוגמא עמודת סיסמת הכניסה לאתר. המטרה העסקית היא לתת הרשאות צפייה מתאימות לעובדים השונים. נרצה לאפשר גישה לטבלת המשתמשים למספר גורמים בארגון לצורך ניתוח נתונים, אולם לא נרצה שכולם יהיו חשופים לסיסמאות של המשתמשים.

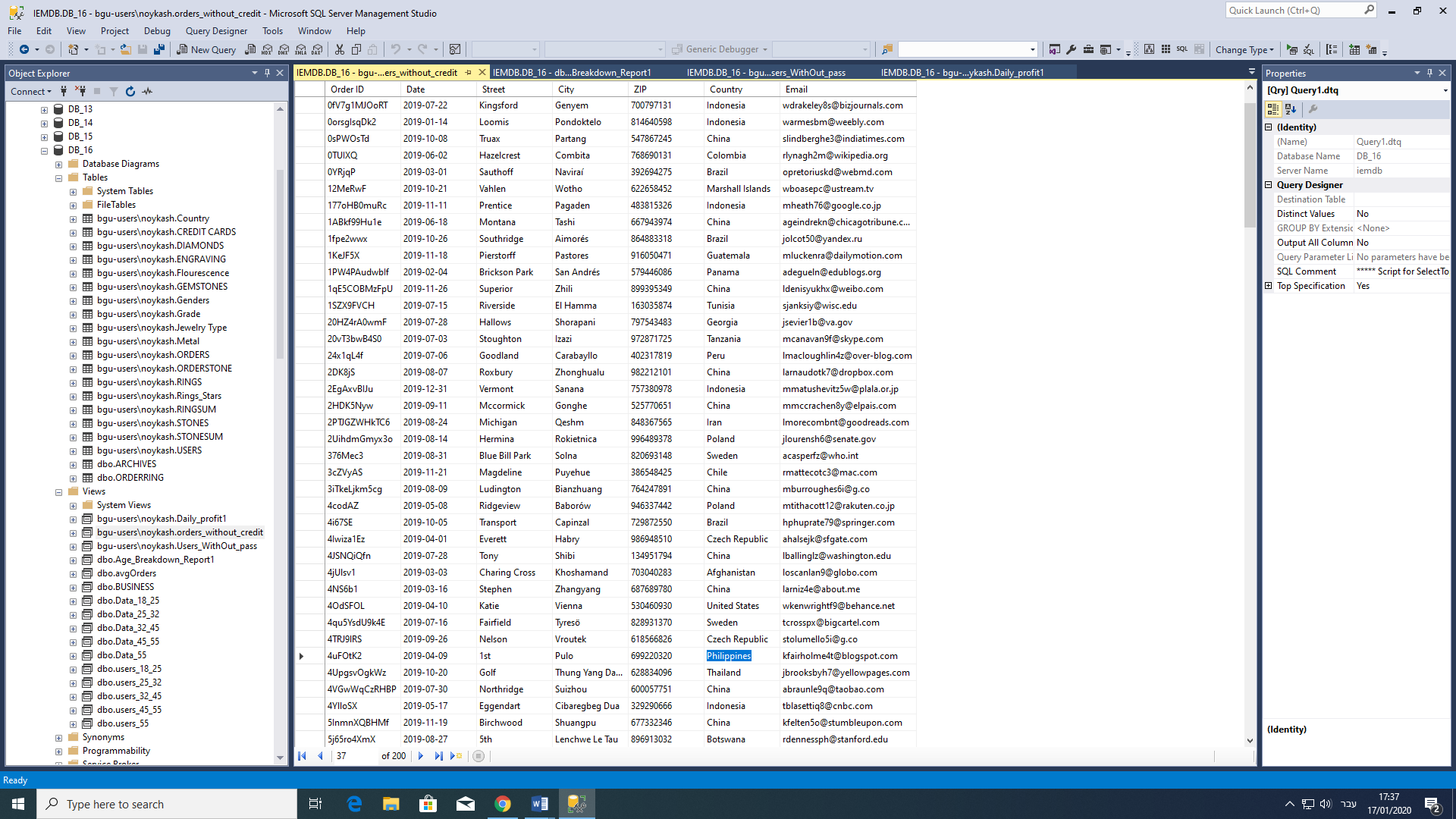
create view orders\_without\_credit

as

SELECT o.[Order ID] , o.Date , o.Street , o.City , o.ZIP , o.Country , o.Email

FROM ORDERS as o

**דוגמא לפלט מהSQL-**



# פונקציות (Functions) 10%

**פונקציה שמחזירה טבלה:**

פונקציה המקבלת מייל של לקוח, ומחזירה טבלה המסכמת את כלל ההזמנות שביצע באתר, ועבור כל הזמנה את סך התשלום. לשם כך, יצרנו שתי טבלאות שבה הורדה רמת הגרעיניות על מנת להימנע מ- DOUBLE COUNTING.

--1 טבלה של הזמנות של טבעות

SELECT ORDERS.[Order ID], Orders.Email, TotalAmountRings= ISNULL(SUM(ROUND(ORDERRING.Quantity\*RINGS.Price + ISNULL(Engraving.Price,0), 2)),0)

INTO RINGSUM

FROM ORDERS LEFT OUTER JOIN ORDERRING ON ORDERS.[Order ID]=ORDERRING.[Order ID] LEFT OUTER JOIN RINGS ON ORDERRING.[Ring ID]=RINGS.[Item Number] LEFT OUTER JOIN ENGRAVING ON RINGS.Engraving=ENGRAVING.Characters

GROUP BY ORDERS.[Order ID], Orders.Email

--2 טבלה של הזמנות של אבנים

SELECT ORDERS.[Order ID], TotalAmountStones = ISNULL(SUM(ROUND(ORDERSTONE.Quantity\*STONES.Price, 2)),0)

INTO STONESUM

FROM ORDERS LEFT OUTER JOIN ORDERSTONE ON ORDERS.[Order ID]=ORDERSTONE.[Order ID] LEFT OUTER JOIN STONES ON ORDERSTONE.[Stone ID]=STONES.[Item Number]

GROUP BY ORDERS.[Order ID]

--הפונקציה

GO

CREATE FUNCTION OrderTotalPrice(@CostumerEmail Varchar(50))

RETURNS TABLE

RETURN (

SELECT R.[Order ID], TotalPrice = r.TotalAmountRings + s.TotalAmountStones

FROM RINGSUM AS R FULL OUTER JOIN STONESUM AS S ON R.[Order ID]= S.[Order ID]

WHERE R.Email = @CostumerEmail

)

GO

**הפעלה של הפונקציה:**

select \*

from OrderTotalPrice('aandrzej67@ft.com')

**פונקציה שמחזירה סקלר:**

פונקציה אשר מקבלת מייל של לקוח ומחזירה את הדירוג שלו עפ"י סך ההכנסות שלו לאתר. לשם כך, השתמשנו בפונקציה הקודמת.

GO

CREATE FUNCTION CostumerRank (@CostumerEmail Varchar(50))

RETURNS CHAR

AS BEGIN DECLARE @Rank CHAR, @Total int

SELECT @Total = (select SUM(TotalPrice) from [bgu-users\noykash].OrderTotalPrice(@CostumerEmail)),

@Rank= CASE WHEN @Total < 10000 THEN 'C'

WHEN @Total BETWEEN 10000 AND 50000 THEN 'B'

ELSE 'A'

END

RETURN @Rank

END

# Trigger פשוט 5%

ברגע שמתווספת טבעת חדשה להזמנה, הטריגר מחשב את ממוצע הכוכבים החדש עבור הטבעת בטבלת הטבעות.

דירוג טבעות מתבצע בתדירות גבוהה כי כל לקוח חייב לדרג את הטבעת שהוא קנה, ונרצה כי ממוצע הכוכבים לטבעת יהיה עדכני ככל האפשר. לכן נרצה שבאופן אוטומטי ממוצע הכוכבים יחושב ברגע שיש הזמנה חדשה.

--יצירת טבלה חדשה: טבלת דירוגים

create table Rings\_Stars(

Email varchar(50) not null ,

Item\_Number varchar(20) not null ,

Number\_Of\_Stars int not null,

constraint pk\_Stars PRIMARY KEY (Email, Item\_Number, Number\_Of\_Stars),

constraint fk\_Stars\_Email FOREIGN KEY (Email) references USERS (Email),

constraint fk\_Stars\_ItemNum FOREIGN KEY (Item\_Number) references RINGS ([Item Number]),

constraint cc\_num CHECK (Number\_Of\_Stars >= 1 and Number\_Of\_Stars<=5),)

--יצירת השדה המחושב:

ALTER TABLE RINGS

ADD Stars\_Avg float

--חישוב הערכים בשדה עבור כל טבעת

UPDATE RINGS

SET Stars\_Avg= (

SELECT Avg(cast(Number\_Of\_Stars as float))

FROM Rings\_Stars as RS

WHERE RS.Item\_Number = RINGS.[Item Number])

--בדיקה רנדומלית

select \*

from RINGS

where [Item Number] = 'NMiLUePzoT9K'

--יצירת הטריגר

CREATE TRIGGER Calculate\_Average\_Stars\_For\_Rings

ON Rings\_Stars

AFTER INSERT

AS

UPDATE RINGS

SET Stars\_Avg= (

SELECT Avg(cast(Number\_Of\_Stars as float))

FROM Rings\_Stars as RS

WHERE RS.Item\_Number in (SELECT Item\_Number FROM INSERTED))

# פרוצדורה שמורה ((Stored Procedure פשוטה 5%

הפרוצדורה מקבלת את כל השדות שיחד יכולים לזהות באופן חד-חד ערכי טבעת ספציפית.

הפרוצדורה עוזרת לעובדי החברה(המורשים לעשות זאת) לעדכן את מלאי המוצרים ע"י הוספה של טבעות חדשות למלאי ללא צורך בהבנה של SQL ושינוים בכתיבת קוד.

Create procedure dbo.Add\_New\_Ring

--הפרוצדורה מקבלת מידע בסיסי על טבעת

@Item\_number varchar(20) , @Jewelry\_Type varchar(16) , @Metal varchar(20) ,  
@weight real, @Side\_Stones varchar(50) , @price real , @Stone\_Number varchar(20),

@Engraving varchar(100), @Engraving\_Price real

As

insert into [bgu-users\noykash].RINGS([Item Number],[Jewelry Type], Metal, [Side Stones],

Weight, Price, [Stone Number], Engraving)

values ( @Item\_number , @Jewelry\_Type , @Metal , @weight, @Side\_Stones , @price ,

@Stone\_Number , @Engraving)

--עידכון טבלת חריטה

insert into ENGRAVING (Characters,Price)

values (@Engraving, @Engraving\_Price)  
 --סוף הפרוצדורה השלמה

* להדגמה ראויה של הפרוצדורה - יש לכתוב שורת ביצוע (Execute) המפעילה אותה. במפגש ההגנה תידרשו להמחיש גם את תוצאות הפעלת הפרוצדורה – למשל, על ידי הצגת הנתונים הרלוונטיים לפני ואחרי פעולתה.

## מטלה 3 (20%) – כלים להצגת נתונים

**דו"ח עסקי (10%) –**

דו"ח זה מיועד למחלקת השיווק של האתר. מחלקת השיווק עוסקת בשיווק ופרסום האתר ברחבי העולם. מטרתה להגדיל את כמות המשתמשים באתר במדינות שונות ברחבי העולם ולהבין באילו מדינות יש צורך לפעול באופן חזק יותר, זאת על ידי התאמת המוצרים לקהל היעד. לכן, מסנני הדו"ח הם עפ"י מדינות.

CREATE VIEW BUSINESS AS

SELECT T1.COUNTRY, [Number Of Users], [Number Of Registered], [Most Popular Ring], [Most Popular Diamond], [Most Popular Gemstone], [Average Num of Orders], [Num of Orders], [Average Age], [Total Sales]

FROM

(

-- התפלגות משתמשים רשומים ולא רשומים

SELECT C.Country, [Number Of Users] = COUNT(Email)

FROM [bgu-users\noykash].Country AS C LEFT OUTER JOIN [bgu-users\noykash].USERS AS U ON C.Country=U.Country

GROUP BY C.Country

) AS T1 JOIN

(

-- התפלגות משתמשים רשומים בלבד

SELECT T1.Country, [Number Of Registered]=ISNULL([Number Of Registered],0)

FROM [bgu-users\noykash].Country AS T1 LEFT OUTER JOIN (

SELECT C.Country, [Number Of Registered] = COUNT(C.Country)

FROM [bgu-users\noykash].Country AS C LEFT OUTER JOIN [bgu-users\noykash].USERS AS U ON C.Country=U.Country

WHERE Registerd=1

GROUP BY C.Country) AS T2 ON T1.Country=T2.Country

) AS T2 ON T1.Country=T2.Country JOIN

(

--סוג הטבעת הנמכר ביותר

select Country , [Most Popular Ring] = ISNULL((SELECT top 1 [Jewelry Type]

FROM ORDERRING JOIN [bgu-users\noykash].RINGS ON ORDERRING.[Ring ID] = RINGS.[Item Number] join [bgu-users\noykash].ORDERS as o1 on o1.[Order ID]=ORDERRING.[Order ID]

where P.country = country

GROUP BY [Jewelry Type]

ORDER BY count([Jewelry Type]) DESC),'NONE')

from [bgu-users\noykash].Country as P

) AS T3 ON T1.Country=T3.Country JOIN

(

--צורת היהלום הנמכרת ביותר

select Country , [Most Popular Diamond] = ISNULL((SELECT top 1 Shape

FROM [bgu-users\noykash].ORDERSTONE as c JOIN [bgu-users\noykash].STONES as q ON c.[Stone ID] = q.[Item Number] join [bgu-users\noykash].ORDERS as o1 on o1.[Order ID]= c.[Order ID] join [bgu-users\noykash].DIAMONDS as d on d.[Item Number] = q.[Item Number]

where P.country = country

GROUP BY Shape

ORDER BY count(Shape) DESC),'NONE')

from [bgu-users\noykash].Country as P

) AS T4 ON T1.Country=T4.Country JOIN

(

--צורת אבן החן הנמכרת ביותר

select Country , [Most Popular Gemstone] = ISNULL((SELECT top 1 Shape

FROM [bgu-users\noykash].ORDERSTONE as c JOIN [bgu-users\noykash].STONES as q ON c.[Stone ID] = q.[Item Number] join [bgu-users\noykash].ORDERS as o1 on o1.[Order ID]= c.[Order ID] join [bgu-users\noykash].GEMSTONES as d on d.[Item Number] = q.[Item Number]

where P.country = country

GROUP BY Shape

ORDER BY count(Shape) DESC),'NONE')

from [bgu-users\noykash].Country as P

) AS T5 ON T1.Country=T5.Country JOIN

(

--ממוצע מספר ההזמנות עבור כל מדינה

Select country, [Average Num of Orders] =

Case when CAST((select COUNT(\*)

from [bgu-users\noykash].USERS

where Country = C.Country) AS FLOAT) > 0 THEN (

CAST((select Count(\*)

from [bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email

where q.Country = C.Country) AS float)

/

CAST((select COUNT(\*)

from [bgu-users\noykash].USERS

where Country = C.Country) AS FLOAT ))

else 0

END

from [bgu-users\noykash].Country as C

) AS T6 ON T1.Country=T6.Country JOIN

(

--סך ההזמנות עבור כל מדינה

Select country, [Num of Orders] =

(select Count(\*)

from [bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email

where q.Country = C.Country)

from [bgu-users\noykash].Country as C

) AS T7 ON T1.Country=T7.Country JOIN

(

--ממוצע גילאים עבור כל מדינה

Select country, [Average Age] = ISNULL(YEAR(GETDATE()) -

(select AVG(YEAR([Birth date]))

from [bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email

where q.Country = C.Country),0)

from [bgu-users\noykash].Country as C

) AS T8 ON T1.Country=T8.Country JOIN

(

--סכום מכירות עבור כל מדינה

select country , [Total Sales] = ISNULL(( select Sum([total sum])

from ( select w.Email , [total sum] = (select [Total Price] = SUM(TotalPrice)

from [bgu-users\noykash].OrderTotalPrice(w.Email))

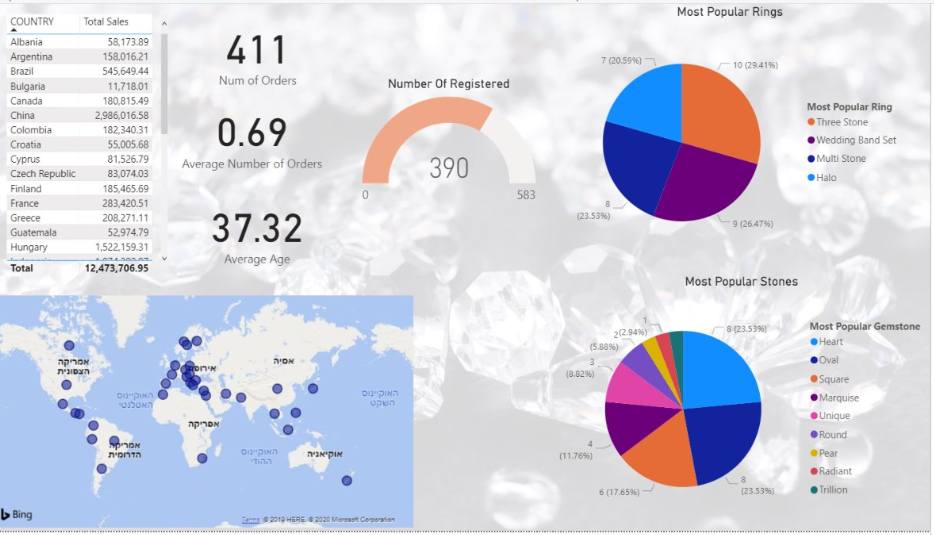
from ([bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email)

where q.Country = C.Country

Group by w.Email ) AS T ) , 0 )

from [bgu-users\noykash].Country as C

) AS T9 ON T1.Country=T9.Country



**התרשימים בדו"ח:**

* טבלה המציגה עבור כל מדינה את סך המכירות.
* עבור כל מדינה, מספר ההזמנות הכולל.
* עבור כל מדינה, הגיל הממוצע של משתמשים אשר ביצעו הזמנות.
* עבור כל מדינה, מהו מספר ההזמנות הממוצע למשתמש.
* שעון המציג את מספר המשתמשים הרשומים מתוך כלל המשתמשים המבצעים הזמנות באתר.
* גרף עוגה המציג את הטבעת הפופולרית ביותר.
* גרף עוגה המציג את האבן הפופולרית ביותר (יהלום ואבן חן יחדיו), כאשר הסינון נעשה על פי צורת האבן.

**לוח מחוונים (10%) עבור המנהלים הבכירים בחברה -**

סך הכנסות בכל הזמנים:

Create view [total income] as

select [Total sum] = Sum([total sum])

from ( select w.Email , [total sum] = (select [Total Price] = SUM(TotalPrice)

from [bgu-users\noykash].OrderTotalPrice(w.Email))

from ([bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email)

Group by w.Email ) AS T

התפלגות מזמינים לפי מין:

Create view [gender percentage] as

select Gender , [Gender Percentage] = (100 \*

cast((select count(Gender)

from [bgu-users\noykash].ORDERS as o join [bgu-users\noykash].USERS as u on o.Email = u.Email

where C.Gender = Gender

group by Gender) as float)

/

cast(( select count(\*)

from [bgu-users\noykash].ORDERS as o join [bgu-users\noykash].USERS as u on o.Email = u.Email )as float))

from [bgu-users\noykash].Genders as C

טופ 5 קונים בכל הזמנים:

Create view [top 5] as

select top 5 w.Email , [Total sum] = (select [Total Price] = SUM(TotalPrice)

from [bgu-users\noykash].OrderTotalPrice(w.Email))

from ([bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email)

Group by w.Email

order by [total sum] desc

גיל ממוצע של המזמינים:

Create view [Average age] as

Select [average Age] = YEAR(GETDATE()) -

(select AVG(YEAR([Birth date]))

from [bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email)

מספר הזמנות לפי שנה:

Create view [ordes num by year]as

Select years , [Num of Orders] =

(select Count(\*)

from [bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email

where year(date) = C.[years])

from (select years = year(date)

from [bgu-users\noykash].ORDERS

group by year(date)) as C

סכום מכירות לפי שנים:

Create view [ordes sum by year] as

select [year] , [Total Sales] = ISNULL(( select Sum([total sum])

from ( select w.Email , [total sum] = (select [Total Price] = SUM(TotalPrice)

from [bgu-users\noykash].OrderTotalPrice(w.Email))

from ([bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email)

where year(date) = C.[year]

Group by w.Email ) AS T ) , 0 )

from (select [year] = year(date)

from [bgu-users\noykash].ORDERS

group by year(date)) as C

מספר הזמנות לפי חודשים:

Create view [ordes num by month]as

Select [month] , [Num of Orders] =

(select Count(\*)

from [bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email

where month(date) = C.[month])

from (select [month] = month(date)

from [bgu-users\noykash].ORDERS

group by month(date)) as C

סכום מכירות לפי חודשים:

Create view [ordes sum by month] as

select [month] , [Total Sales] = ISNULL(( select Sum([total sum])

from ( select w.Email , [total sum] = (select [Total Price] = SUM(TotalPrice)

from [bgu-users\noykash].OrderTotalPrice(w.Email))

from ([bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email)

where month(date) = C.[month]

Group by w.Email ) AS T ) , 0 )

from (select [month] = month(date)

from [bgu-users\noykash].ORDERS

group by month(date)) as C

אחוז הלקוחות הרשומים שביצעו הזמנה:

Create view [users that order percantage] as

select [user percantage order] = 100\*(

cast((select sum(T)

from (select count(\*) AS T

from [bgu-users\noykash].ORDERS as o join [bgu-users\noykash].USERS as u on o.Email = u .Email

where Registerd = 1

group by u.Email) AS Q) as float)

/

cast((select count(\*)

from [bgu-users\noykash].USERS

where Registerd = 1 ) as float)) , Target = 100

הכנסות מול מטרות לפי כל שנה:

create view [income to target] as

select year , [Total Sales] , [Target]

from [Total income target] as I join ( select [year] , [Total Sales] = ISNULL(( select Sum([total sum])

from ( select w.Email , [total sum] = (select [Total Price] = SUM(TotalPrice)

from [bgu-users\noykash].OrderTotalPrice(w.Email))

from ([bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email)

where year(date) = C.[year]

Group by w.Email ) AS T ) , 0 )

from (select [year] = year(date)

from [bgu-users\noykash].ORDERS

group by year(date)) as C ) AS Q on I.Years = Q.[year]

מספר הזמנות מול מטרות לפי כל שנה:

create view [orders target] as

select [year] , [Target] = [number of orders] , [Num of Orders]

from [order number target] as I join (Select years , [Num of Orders] =

(select Count(\*)

from [bgu-users\noykash].ORDERS as w join [bgu-users\noykash].USERS as q on w.Email = q.Email

where year(date) = C.[years])

from (select years = year(date)

from [bgu-users\noykash].ORDERS

group by year(date)) as C ) AS Q on I.[year] = Q.[years]



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

## כלי מורכב (Stored Procedure או Trigger) 10%

**TRIGGER:**

לצורך דרבון לקוחות לבצע מס' רב של הזמנות, הוספנו את השדה "[Level]" לישות לקוח. לקוח שלא ביצע הזמנה כלל יקבל את הדרגה – bad, לקוח שביצע הזמנות מתחת לממוצע ההזמנות ללקוח יקבל את הדרגה – beginner, לקוח שביצע הזמנות מעל הממוצע יקבל את הדרגה – expert ולקוח שביצע הזמנות בסך שעולה על פי 2 מהממוצע יקבל את הדרגה – master.

הטריגר פועל על טבלת ORDERS ברגע שמתבצעת פעולת INSERT.

הטריגר נעזר בפונקציה המקבלת מייל של לקוח ומחזירה את סך ההוצאות שלו, ב- VIEW המחזיר את ממוצע ההוצאות של כל הלקוחות ובפרוצדורה המקבלת מייל של לקוח ודרגה ומעדכנת בטבלת USERS את הדרגה של הלקוח בעל המייל שהתקבל להיות הדרגה שהתקבלה.

**הוספת השדה -**

alter table USERS

add [Level] varchar(10)

**פונקציה המחזירה את מספר ההזמנות של הלקוח בעל המייל שהתקבל -**

GO

create function dbo.totalorders (@email varchar(50))

returns int

as begin

declare @total int

select @total=count(email)

from orders

where @email= email

return @total

end

GO

**VIEWשל ממוצע מספר ההזמנות של כל הלקוחות -**

create view dbo.avgOrders as

select average = CAST((select count(\*)

from [bgu-users\noykash].ORDERS) AS FLOAT) / CAST((select Count(\*)

from [bgu-users\noykash].USERS) AS FLOAT)

**פרוצדורה המגדירה דרגה ללקוח בהתאם לדרגה והמייל שהתקבלו -**

create procedure dbo.setLevel @email varchar(50), @Level varchar(10)

as

update [bgu-users\noykash].USERS set [Level] = @Level

where email = @email

**טריגר המעדכן את הדרגות של כלל הלקוחות -**

create trigger updatelevel

on orders

for insert as

declare @email varchar(50)

declare @level varchar(10)

declare OurCursor cursor

for select email

from USERS

begin

open OurCursor

fetch next from OurCursor into @email

while (@@FETCH\_STATUS = 0)

begin

if (dbo.totalorders(@email)>(select average from dbo.avgOrders)\*2) begin execute dbo.setLevel @email, 'master' end

else if (dbo.totalorders(@email)>(select average from dbo.avgOrders )) begin execute dbo.setLevel @email, 'expert' end

else if ((dbo.totalorders(@email)<=(select average from dbo.avgOrders )) AND (dbo.totalorders(@email) >0)) begin execute dbo.setLevel @email,'begginer' end

else begin execute dbo.setLevel @email,'bad' end

fetch next from OurCursor

into @email

end

close OurCursor

end

## שילוב מערכתי של מספר כלים 10%

**בסעיף זה השתמשנו בפרוצדורה, פונקציה וטריגר:**

באם משתמש ירצה למחוק את המשתמש שלו, הפונקציה תבדוק קודם כל את מהימנות המשתמש שמבצע את הפעולה. אם תמצא התאמה בין השם ולבין הסיסמה שלו, רשומת המשתמש תמחק את טבלת המשתמשים ותועבר לטבלת "ארכיון". כך, אם המשתמש יחזור בו מהחלטתו של מחיקת המשתמש, פרטיו כבר יהיו שמורים במקום אחר במערכת.

**פרוצדורה שמקבלת שם וססמא ומוחקת את המשתמש המתאים מטבלת USERS:**

CREATE PROCEDURE DeleteUser (@Email varchar(50), @Password varchar(50)) AS

Update [bgu-users\noykash].Orders

set Email = NULL

WHERE dbo.IsCorrect(@Email,@Password) = 1 AND @Email=Orders.[Email]

DELETE [bgu-users\noykash].Rings\_Stars

where dbo.IsCorrect(@Email,@Password) = 1 AND @Email=Rings\_Stars.[Email]

DELETE [bgu-users\noykash].USERS

WHERE dbo.IsCorrect(@Email,@Password) = 1 AND @Email=USERS.[Email]

**דוגמא לשימוש:**

EXECUTE dbo.DeleteUser 'oglaysherb4@europa.eu','U5dC683h'

## פונקציה שבודקת האם הסיסמא תואמת לשם משתמש:

CREATE FUNCTION IsCorrect (@Email varchar(50), @Password varchar(50))

RETURNS INTEGER

AS

BEGIN

DECLARE @answer INTEGER

SELECT @answer= COUNT([Email])

FROM [bgu-users\noykash].USERS

WHERE @Email = Email AND @Password= [Password]

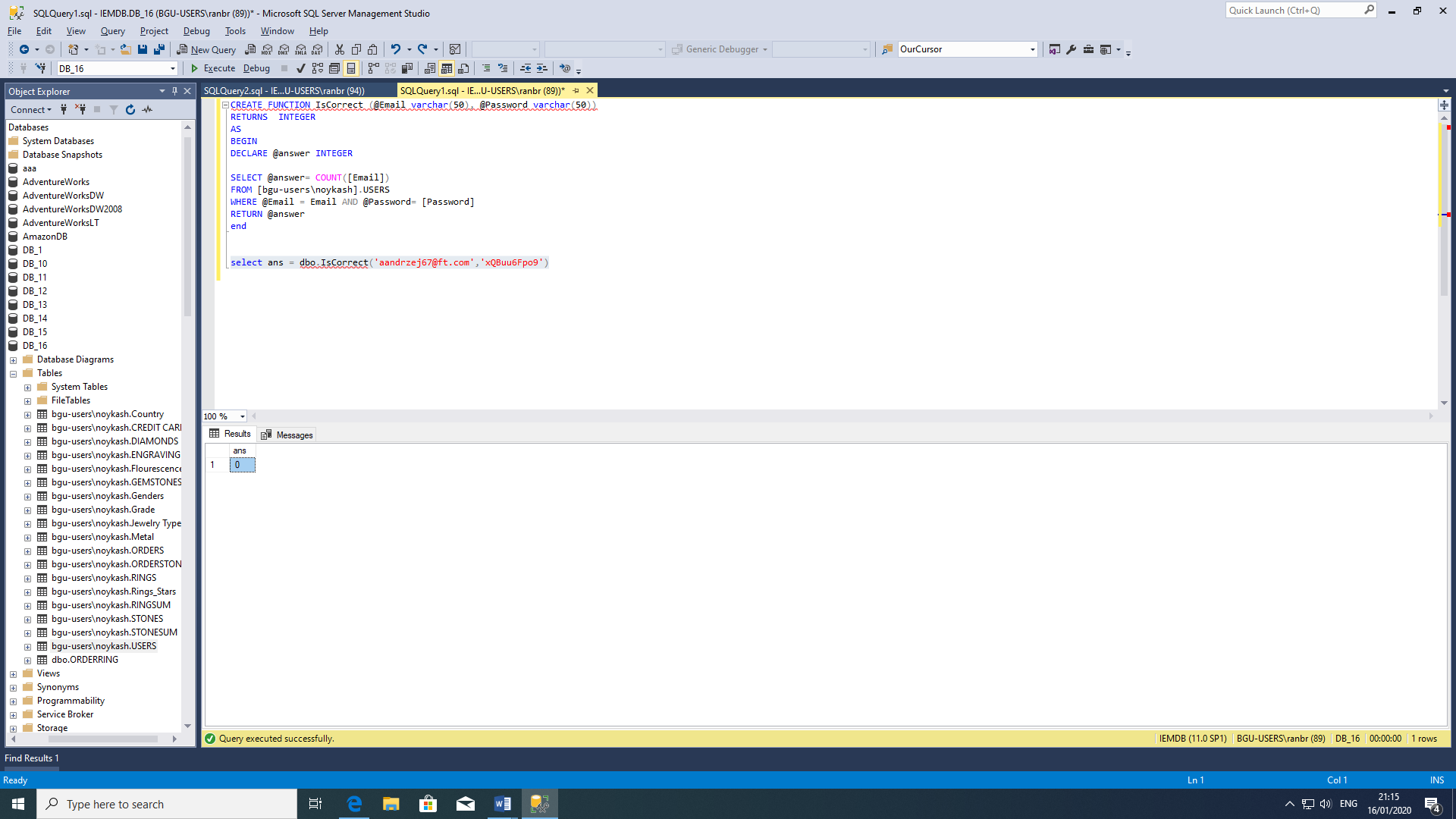
RETURN @answer

end

**דוגמא לשימוש בפונקציה:**

select ans = dbo.IsCorrect('aandrzej67@ft.com','xQBuu6Fpo9')

פלט-



**יצירת טבלת ARCHIVES, אליה מועברים המשתמשים לאחר מחיקה:**

CREATE TABLE ARCHIVES (

Email Varchar(50) not null,

[First Name] Varchar(15) not null,

[Last Name] Varchar(15) not null,

[Phone Number] char(10) not null,

[Birth date] Date not null,

Gender Char(1) not null foreign key references [bgu-users\noykash].Genders(Gender),

Street Varchar(20) null,

City Varchar(20) null,

ZIP Char(9) null,

Country Varchar(100) null foreign key references [bgu- users\noykash].Country (Country),

Registerd bit DEFAULT 0 not null,

Password varchar(50) null,

Level varchar(10) null,

constraint pk\_users567 PRIMARY KEY (Email), --מפתח ראשי

) )

**יצירת טריגר המופעל לאחר מחיקת ערכים מטבלת USERS:**

CREATE TRIGGER AddToArchives

ON [bgu-users\noykash].USERS after DELETE AS INSERT INTO ARCHIVES

SELECT \*

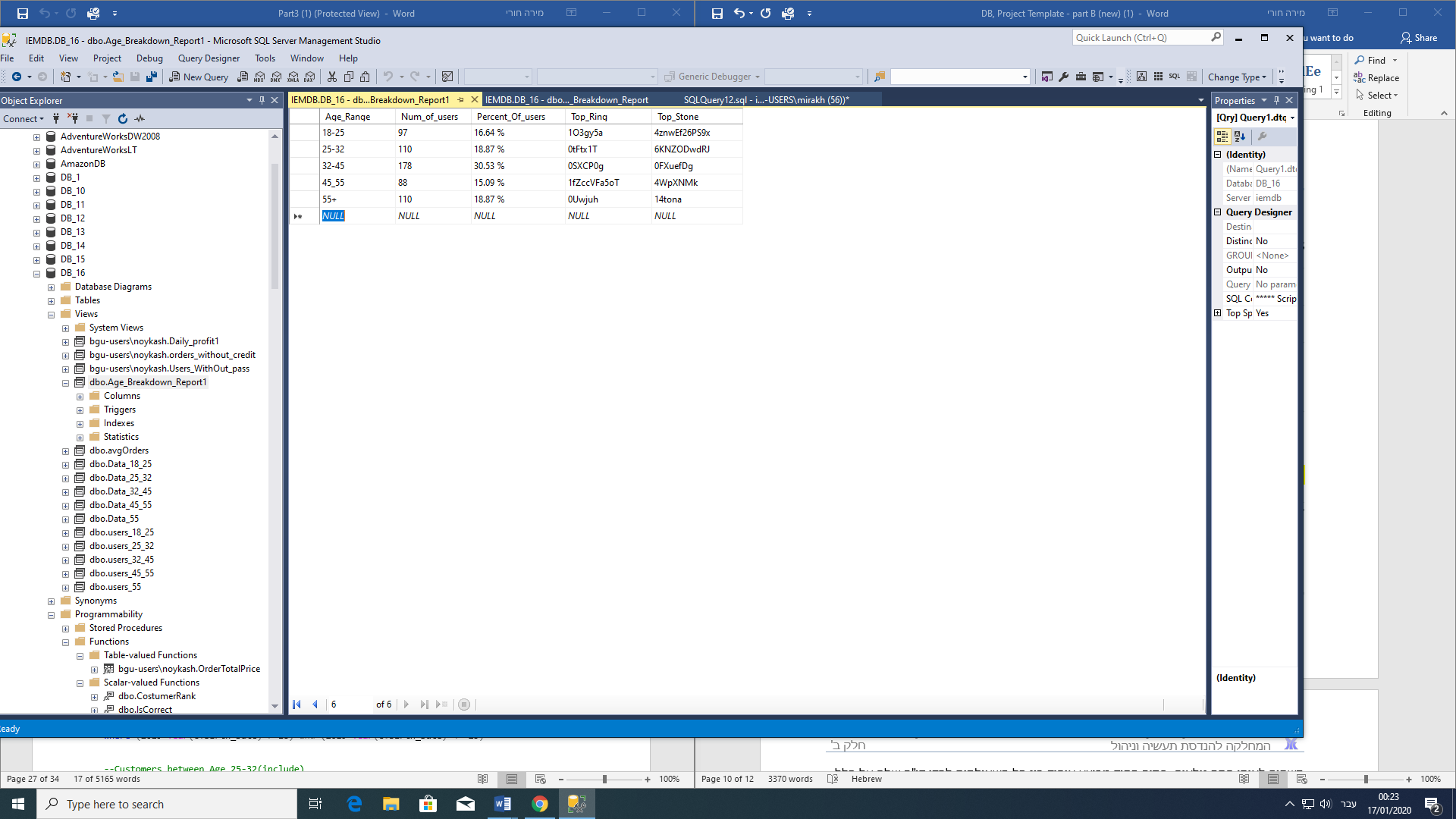
## FROM DELETED

## דו"ח המושתת על שאילתה מקוננת מורכבת 10%

**דו"ח - פילוח גילאי לקוחות**

הדוח מתאר עבור כל חתך גילאים – את מספר הלקוחות מאותו טווח והיחס מתוך כלל הלקוחות, כמו כן – הדוח מתאר את דפוסי הקניה של אותו טווח גילאים – לדוגמה – מה הטבעת הכי נקנית או כמה אחוז מהווה כל חתך גילאים מסך המשתמשים . כמו כן, הדוח נותן תשקיף ועוזר ליישם תוכן שיותר רלוונטי עבור כל חתך גילאים. לטובת הדוח – הופקה טבלת VIEW לכל חתך גילאי לקוחות. עבור כל חתך גילאים נכתבו שתי פונקציות המחזירות סקלר ונכתבה שאילתה מורכבת שמשתמשת באותן פונקציות ונותנת תשקיף לאותו חתך גילאים. בסוף, הקוד מבצע איחוד בין כל השאילתות לכדי דו"ח שלם על כלל חתכי גילאי הלקוחות.

**פלט סופי מהSQL -**



**קוד -**

-- תצוגות בסיסיות של משתמשים לפי גיל

-- (משתמשים בגילאים 18-25 (כולל 25

Create view dbo.users\_18\_25

As

select \*

from [bgu-users\noykash].USERS as U

where (2020-Year(U.[Birth Date]) > 18) and (2020-Year(U.[Birth Date]) <= 25)

--(משתמשים בגילאים 25-32 (כולל 32

Create view dbo.users\_25\_32

As

select \*

from [bgu-users\noykash].USERS as U

where (2020-Year(U.[Birth Date]) > 25) and (2020-Year(U.[Birth Date]) <= 32)

--(משתמשים בגילאים 32-45 (כולל 45

Create view dbo.users\_32\_45

As

select \*

from [bgu-users\noykash].USERS as U

where (2020-Year(U.[Birth Date]) > 32) and (2020-Year(U.[Birth Date]) <= 45)

--(משתמשים בגילאים 45-55 (כולל 55

Create view dbo.users\_45\_55

As

select \*

from [bgu-users\noykash].USERS as U

where (2020-Year(U.[Birth Date]) > 45) and (2020-Year(U.[Birth Date]) <= 55)

--משתמשים מעל גיל 55

Create view dbo.users\_55

As

select \*

from [bgu-users\noykash].USERS as U

where (2020-Year(U.[Birth Date]) > 55)

-------פונקציות וטבלאות ויו עבור טווח הגילאים: 18-25 ------

--חישוב אחוז המשתמשים בגילאי 18-25 מתוך כלל המשתמשים

GO

Create function dbo.Percent\_of\_users\_18\_25()

Returns DOUBLE PRECISION

As

Begin

Declare @total DOUBLE PRECISION

Declare @aged DOUBLE PRECISION

set @total = (select count(\*) as float from [bgu-users\noykash].USERS)

set @aged = (select count(\*) from dbo.users\_18\_25)

Return @aged/@total

End

GO

--חישוב אחוז ההזמנות שנעשו ע"י משתמשים בגילאי 18-25 מתוך כלל הזמנות המשתמשים

GO

Create function dbo.Percent\_Bought\_18\_25()

Returns DOUBLE PRECISION

As

Begin

Declare @total\_bought DOUBLE PRECISION

Declare @bought DOUBLE PRECISION

set @bought = (select count(\*)

from users\_18\_25 )

set @total\_bought = (select count(\*) From [bgu-users\noykash].ORDERS )

Return @bought/@total\_bought

End

GO

--view

Create view dbo.Data\_18\_25

as

select [Age\_Range] = ('18-25') ,

[Num\_of\_users] = (select count(\*) from dbo.users\_18\_25) ,

[Percent\_Of\_users] = (select Cast(Cast((select dbo.Percent\_of\_users\_18\_25())\*100 as

decimal(18,2)) as varchar(5)) + ' %' as Percentage),

[Top\_Ring] = (select top 1 [Item Number]

from [bgu-users\noykash].RINGS as r JOIN dbo.ORDERRING as Oring on

r.[Item Number]=Oring.[Ring ID] JOIN [bgu-users\noykash].ORDERS as o on o.[Order ID]=Oring.[Order ID]JOIN dbo.users\_18\_25 as u on o.Email=u.Email

group by [Item Number]

order by Count([Item Number]) desc ),

[Top\_Stone] = (select top 1 [Item Number]

from [bgu-users\noykash].STONES as r JOIN [bgu-users\noykash].ORDERSTONE as

O on r.[Item Number]=O.[STONE ID] JOIN [bgu-users\noykash].ORDERS as o1 on o1.[Order ID]=O.[Order ID] JOIN dbo.users\_18\_25 as u on o1.Email=u.Email

group by [Item Number]

order by Count([Item Number]) desc )

------- פונקציות וטבלאות ויו עבור טווח הגילאים: 25-32 ------

--חישוב אחוז המשתמשים בגילאי 25-32 מתוך כלל המשתמשים

GO

Create function dbo.Percent\_of\_users\_25\_32 ()

Returns DOUBLE PRECISION

As

Begin

Declare @total DOUBLE PRECISION

Declare @aged DOUBLE PRECISION

set @total = (select count(\*) as float from [bgu-users\noykash].USERS)

set @aged = (select count(\*) from dbo.users\_25\_32)

Return @aged/@total

End

GO

--חישוב אחוז ההזמנות שנעשו ע"י משתמשים בגילאי 25-32 מתוך כלל הזמנות המשתמשים

GO

Create function dbo.Percent\_Bought\_25\_32 ()

Returns DOUBLE PRECISION

As

Begin

Declare @total\_bought DOUBLE PRECISION

Declare @bought DOUBLE PRECISION

set @bought = (select count(\*)

from users\_25\_32)

set @total\_bought = (select count(\*) From [bgu-users\noykash].ORDERS )

Return @bought/@total\_bought

End

GO

--view

Create view dbo.Data\_25\_32

as

select [Age\_Range] = ('25-32') ,

[Num\_of\_users] = (select count(\*) from dbo.users\_25\_32) ,

[Percent\_Of\_users] = (select Cast(Cast((select dbo.Percent\_of\_users\_25\_32())\*100 as

decimal(18,2)) as varchar(5)) + ' %' as Percentage),

[Top\_Ring] = (select top 1 [Item Number]

from [bgu-users\noykash].RINGS as r JOIN dbo.ORDERRING as Oring on

r.[Item Number]=Oring.[Ring ID] JOIN [bgu-users\noykash].ORDERS as o on o.[Order ID]=Oring.[Order ID]

JOIN dbo.users\_25\_32 as u on o.Email=u.Email

group by [Item Number]

order by Count([Item Number]) desc ),

[Top\_Stone] = (select top 1 [Item Number]

From[bgu-users\noykash].STONES as r JOIN [bgu-

users\noykash].ORDERSTONE as O on r.[Item Number]=O.[STONE ID]

JOIN [bgu-users\noykash].ORDERS as o1 on o1.[Order ID]=O.[Order ID]

JOIN dbo.users\_25\_32 as u on o1.Email=u.Email

group by [Item Number]

order by Count([Item Number]) desc )

------- פונקציות וטבלאות ויו עבור טווח הגילאים: 32-45 ------

-- חישוב אחוז המשתמשים בגילאי 32-45 מתוך כלל המשתמשים

GO

Create function dbo.Percent\_of\_users\_32\_45 ()

Returns DOUBLE PRECISION

As

Begin

Declare @total DOUBLE PRECISION

Declare @aged DOUBLE PRECISION

set @total = (select count(\*) as float from [bgu-users\noykash].USERS)

set @aged = (select count(\*) from dbo.users\_32\_45)

Return @aged/@total

End

GO

--חישוב אחוז ההזמנות שנעשו ע"י משתמשים בגילאי 32-45 מתוך כלל הזמנות המשתמשים

GO

Create function dbo.Percent\_Bought\_32\_45 ()

Returns DOUBLE PRECISION

As

Begin

Declare @total\_bought DOUBLE PRECISION

Declare @bought DOUBLE PRECISION

set @bought = (select count(\*)

from users\_32\_45)

set @total\_bought = (select count(\*) From [bgu-users\noykash].ORDERS )

Return @bought/@total\_bought

End

GO

--view

Create view dbo.Data\_32\_45

as

select [Age\_Range] = ('32-45') ,

[Num\_of\_users] = (select count(\*) from dbo.users\_32\_45) ,

[Percent\_Of\_users] = (select Cast(Cast((select dbo.Percent\_of\_users\_32\_45())\*100 as

decimal(18,2)) as varchar(5)) + ' %' as Percentage),

[Top\_Ring] = (select top 1 [Item Number]

from [bgu-users\noykash].RINGS as r JOIN dbo.ORDERRING as Oring on

r.[Item-Number]=Oring.[Ring ID] JOIN [bgu-users\noykash].ORDERS as o on o.[Order ID]=Oring.[Order ID] JOIN dbo.users\_32\_45 as u on o.Email=u.Email

group by [Item Number]

order by Count([Item Number]) desc ),

[Top\_Stone] = (select top 1 [Item Number]

from [bgu-users\noykash].STONES as r JOIN [bgu-users\noykash].ORDERSTONE as

O on r.[Item Number]=O.[STONE ID] JOIN [bgu-users\noykash].ORDERS as o1 on o1.[Order ID]=O.[Order ID]

JOIN dbo.users\_32\_45 as u on o1.Email=u.Email

group by [Item Number]

order by Count([Item Number]) desc )

------- פונקציות וטבלאות ויו עבור טווח הגילאים: 45-55 ------

--חישוב אחוז המשתמשים בגילאי 45-55 מתוך כלל המשתמשים

GO

Create function dbo.Percent\_of\_users\_45\_55 ()

Returns DOUBLE PRECISION

As

Begin

Declare @total DOUBLE PRECISION

Declare @aged DOUBLE PRECISION

set @total = (select count(\*) as float from [bgu-users\noykash].USERS)

set @aged = (select count(\*) from dbo.users\_45\_55)

Return @aged/@total

End

GO

--חישוב אחוז ההזמנות שנעשו ע"י משתמשים בגילאי 45-55 מתוך כלל הזמנות המשתמשים

GO

Create function dbo.Percent\_Bought\_45\_55 ()

Returns DOUBLE PRECISION

As

Begin

Declare @total\_bought DOUBLE PRECISION

Declare @bought DOUBLE PRECISION

set @bought = (select count(\*)

from users\_45\_55)

set @total\_bought = (select count(\*) From [bgu-users\noykash].ORDERS )

Return @bought/@total\_bought

End

GO

--view

Create view dbo.Data\_45\_55

as

select [Age\_Range] = ('45\_55 ') ,

[Num\_of\_users] = (select count(\*) from dbo.users\_45\_55) ,

[Percent\_Of\_users] = (select Cast(Cast((select dbo.Percent\_of\_users\_45\_55 ())\*100 as

decimal(18,2)) as varchar(5)) + ' %' as Percentage),

[Top\_Ring] = (select top 1 [Item Number]

from [bgu-users\noykash].RINGS as r JOIN dbo.ORDERRING as Oring on r.[Item

Number]=Oring.[Ring ID] JOIN [bgu-users\noykash].ORDERS as o on o.[Order ID]=Oring.[Order ID]JOIN dbo.users\_45\_55 as u on o.Email=u.Email group by [Item Number]

order by Count([Item Number]) desc ),

[Top\_Stone] = (select top 1 [Item Number]

from [bgu-users\noykash].STONES as r JOIN [bgu-users\noykash]

.ORDERSTONE as O on r.[Item Number]=O.[STONE ID] JOIN [bgu-users\noykash].ORDERS as o1 on o1.[Order ID]=O.[Order ID]

JOIN dbo.users\_45\_55 as u on o1.Email=u.Email group by [Item Number]

order by Count([Item Number]) desc )

------- פונקציות וטבלאות ויו עבור הגילאים שמעל 55 ------

--חישוב אחוז המשתמשים הגילאים שמעל 55 מתוך כלל המשתמשים

GO

Create function dbo.Percent\_of\_users\_55 ()

Returns DOUBLE PRECISION

As

Begin

Declare @total DOUBLE PRECISION

Declare @aged DOUBLE PRECISION

set @total = (select count(\*) as float from [bgu-users\noykash].USERS)

set @aged = (select count(\*) from dbo.users\_55)

Return @aged/@total

End

GO

--חישוב אחוז ההזמנות שנעשו ע"י משתמשים בגילאים שמעל 55 מתוך כלל הזמנות המשתמשים

GO

Create function dbo.Percent\_Bought\_55 ()

Returns DOUBLE PRECISION

As

Begin

Declare @total\_bought DOUBLE PRECISION

Declare @bought DOUBLE PRECISION

set @bought = (select count(\*)

from users\_55)

set @total\_bought = (select count(\*) From [bgu-users\noykash].ORDERS )

Return @bought/@total\_bought

End

GO

--view

Create view dbo.Data\_55

as

select [Age\_Range] = ('55+') ,

[Num\_of\_users] = (select count(\*) from dbo.users\_55) ,

[Percent\_Of\_users] = (select Cast(Cast((select dbo.Percent\_of\_users\_55())\*100 as

decimal(18,2)) as varchar(5)) + ' %' as Percentage),

[Top\_Ring] = (select top 1 [Item Number]

from [bgu-users\noykash].RINGS as r JOIN dbo.ORDERRING as Oring on

r.[Item Number]=Oring.[Ring ID] JOIN [bgu-users\noykash].ORDERS as o on o.[Order ID]=Oring.[Order ID] JOIN dbo.users\_55 as u on o.Email= u.Email

group by [Item Number]

order by Count([Item Number]) desc ),

[Top\_Stone] = (select top 1 [Item Number]

from [bgu-users\noykash].STONES as r JOIN [bgu-users\noykash].ORDERSTONE

as O on r.[Item Number]=O.[STONE ID] JOIN [bgu-users\noykash].ORDERS as o1 on o1.[Order ID]=O.[Order ID] JOIN dbo.users\_55 as u on o1.Email=u.Email

group by [Item Number]

order by Count([Item Number]) desc )

איחוד כלל הviews כתוצאה מהשאילתות המורכבות של כל חתך לכדי דו"ח:

--איחוד כלל הוויוז כתוצאה מהשאילתות המורכבות של כל חתך לכדי דו"ח:

Create view dbo.Age\_Breakdown\_Report1 as

(select \*

from dbo.Data\_18\_25

UNION

select \*

from dbo.Data\_25\_32

UNION

select \*

from dbo.Data\_32\_45

UNION

select \*

from dbo.Data\_45\_55

UNION

select \*

from dbo.Data\_55)

## נספחים:

## קוד מתוקן ליצירת טבלאות:

CREATE TABLE [CREDIT CARDS] (

[Credit Number] varchar(20) not null,

[Card Holder Name] Varchar(30) not null,

Expiration Date not null,

CVV Char(3) not null,

Type varchar(20) not null,

constraint pk\_creditcards PRIMARY KEY ([Credit Number]),

constraint check\_exp\_type CHECK ([Expiration] > getdate()),

constraint check\_cvv CHECK (CVV LIKE '[0-9][0-9][0-9]'),

)

CREATE TABLE Genders( --LookUp table - #0.1

Gender char(1) not null primary key,

--lookup table for 'gender' in Customers table

)

insert into Genders (Gender) values ('F'),('M')

CREATE TABLE Country ( --LookUp table - #0.2

Country varchar(100) not null primary key,

--lookup table for ' Country ' in USERS table

)

INSERT INTO Country (Country) VALUES

('Afghanistan'),

('Aland Islands'),

('Albania'),

('Algeria'),

('American Samoa'),

('Andorra'),

('Angola'),

('Anguilla'),

('Antartica'),

('Antigua and Barbuda'),

('Argentina'),

('Armenia'),

('Aruba'),

('Australia'),

('Austria'),

('Azerbaijan'),

('Bahamas'),

('Bahrin'),

('Bangladish'),

('Barbados'),

('Belarus'),

('Belguim'),

('Belize'),

('Benin'),

('Bermuda'),

('Bhutan'),

('Bolivia'),

('Bonsia and Herzegovina'),

('Botswana'),

('Bouvet Island'),

('Brazil'),

('British Indian Ocean Territory'),

('Brunei Darussalam'),

('Bulgaria'),

('Burkina Faso'),

('Burundi'),

('Cambodia'),

('Cameroon'),

('Canada'),

('Cape Verde'),

('Cayman Islands'),

('Centeral African Republic'),

('Chad'),

('Chile'),

('China'),

('Christmas Island'),

('Cocos (Keeling) Islands'),

('Colombia'),

('Comoros'),

('Congo'),

('Cook Islands'),

('Costa Rica'),

('Cote Dlvoire'),

('Croatia'),

('Cuba'),

('Cyprus'),

('Czech Republic'),

('Ecuador'),

('Egybt'),

('El Salvador'),

('Equatorial Guinea'),

('Eritrea'),

('Estonia'),

('Ethopia'),

('Falklad Islands(Malvinas)'),

('Faroe Islands'),

('Fiji'),

('Finland'),

('France'),

('French Guiana'),

('French Polynesia'),

('French Southern Territories'),

('Gabon'),

('Gambia'),

('Georgia'),

('Ghana'),

('Gibraltar'),

('Greece'),

('Greenland'),

('Grenada'),

('Guadeloupe'),

('Guam'),

('Guatemala'),

('Guernsey'),

('Guinea'),

('Guinea-Bissau'),

('Guyana'),

('Haiti'),

('Heard Island and McDonald Islands'),

('Holy See (Vatican City State)'),

('Honduras'),

('Hong Kong'),

('Hungary'),

('Iceland'),

('India'),

('Indonesia'),

('Iran'),

('Iraq'),

('Isle of Man'),

('Israel'),

('Italy'),

('Jamaica'),

('Japan'),

('Jersy'),

('Jordan'),

('Kazakhstan'),

('Kenya'),

('Kiribati'),

('Korea, Democratic' ),

('Korea'),

('Kuwait'),

('Kyrgyzstan'),

('Lao Peoples Democratic Republic'),

('Latvia'),

('Lebanon'),

('Lesotho'),

('Liberia'),

('Libyan Arab Jamahiriya'),

('Liechtenstein'),

('Lethuania'),

('Lexumbourg'),

('Macao'),

('Macdonia'),

('Madagascar'),

('Malawi'),

('Malaysia'),

('Maldives'),

('Mali'),

('Malta'),

('Marshall Islands'),

('Matinique'),

('Mauritania'),

('Mauritius'),

('Mayotte'),

('Mexico'),

('Micronesia'),

('Moldova'),

('Monaco'),

('Mongolia'),

('Montenegro'),

('Montserrat'),

('Morocco'),

('Mozambique'),

('Myanmar'),

('Namibia'),

('Nauru'),

('Nepal'),

('Netherlands'),

('Netherlands Antilles'),

('New Caledonia'),

('New Zealand'),

('Nicaragua'),

('Niger'),

('Niue'),

('Norfolk Island'),

('Northen Mariana Islands'),

('Norway'),

('Oman'),

('Pakistan'),

('Paulue'),

('Palestinian Territory, Occupied'),

('Panama'),

('Papua New Guinea'),

('Paraguay'),

('Peru'),

('Philippines'),

('Pitcairn'),

('Poland'),

('Portogal'),

('Puerto Rico'),

('Qatar'),

('Reunion'),

('Romania'),

('Russian Federation'),

('Rwanda'),

('Saint Barthelemy'),

('Saint Helena'),

('Saint Kitts and Nevies'),

('Saint Lucia'),

('Saint Martin'),

('Saint Pierre and Miquelon'),

('Saint Vincent and Grenadines'),

('Samoa'),

('South Africa'),

('South Georgia and the South Sandwich Islands'),

('Spain'),

('Sri Lanka'),

('Sudan'),

('Suriname'),

('Svalbard and Jan Mayen'),

('Swaziland'),

('Sweden'),

('Switzerland'),

('Syrian Arab Republic'),

('Taiwan'),

('Tajikistan'),

('Tanzania'),

('Thailand'),

('Timor-Leste'),

('Togo'),

('Tokelau'),

('Tonga'),

('Trinidad and Tobango'),

('Tunisia'),

('Turkey'),

('Turkemenistan'),

('Turks and Caicos Islands'),

('Tuvalu'),

('Oganda'),

('Ukrania'),

('United Arab Emirates'),

('United Kingdom'),

('United States'),

('Ubited States Minor Outlying Islands'),

('Ourguay'),

('Uzbekistan'),

('Vanuatu'),

('Venezuela'),

('Viet Narm'),

('Virgin Islads, British'),

('Virgin Islads, U.S.'),

('Wallis And Futuna'),

('Western Sahara'),

('Yemen'),

('Zambia'),

('Zembabwe')

CREATE TABLE USERS (

Email Varchar(50) not null,

[First Name] Varchar(15) not null,

[Last Name] Varchar(15) not null,

[Phone Number] char(10) not null,

[Birth date] Date not null,

Gender Char(1) not null foreign key references Genders(Gender),

Street Varchar(20) null,

City Varchar(20) null,

ZIP Char(9) null,

Country Varchar(100) null foreign key references Country (Country),

Registerd bit not null,

Password varchar(50) null,

constraint pk\_users PRIMARY KEY (Email), --מפתח ראשי

constraint check\_email CHECK (Email LIKE '%@%.%'),

constraint check\_phone CHECK ([Phone Number] LIKE ('[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')),

constraint check\_zip CHECK (ZIP LIKE ('[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')),

)

CREATE TABLE ORDERS (

[Order ID] Varchar(20) not null,

Date Date not null,

Street Varchar(20) not null,

City Varchar(20) not null,

ZIP char(9) not null,

Country Varchar(100) foreign key references Country(Country) not null,

[Credit Number] varchar(20) null,

Email Varchar(50) null,

constraint pk\_orders PRIMARY KEY ([Order ID]), --מפתח ראשי

constraint fk\_creditcards FOREIGN KEY ([Credit Number]) REFERENCES [CREDIT CARDS] ([Credit Number]), --מפתח זר

constraint fk\_users2 FOREIGN KEY (Email) REFERENCES USERS (Email), --מפתח זר

)

CREATE TABLE [Jewelry Type]( --LookUp table - #0.3

[Jewelry Type] Varchar(16) not null primary key,

--lookup table for '[Jewelry Type]' in RINGS table

)

insert into [Jewelry Type] ([Jewelry Type]) values ('Solitaire'),('Three Stone'),('Multi Stone'),('Wedding Band Set'),('Halo')

CREATE TABLE Metal( --LookUp table - #0.4

Metal Varchar(20) not null primary key

--lookup table for 'Metal' in RINGS table

)

insert into Metal(Metal) values ('White Gold'),('Yellow Gold'),('Platinum')

CREATE TABLE Grade ( --LookUp table - #0.5

grade varchar(9) not null primary key

--lookup table for 'grade' in some tables

)

insert into Grade(grade) values ('Excellent'),('Very Good'),('Good')

CREATE TABLE STONES (

[Item Number] Varchar(20) not null,

Shape Varchar(20) not null,

Weight real not null,

Cut Varchar(9) foreign key references Grade (grade) not null,

Price real not null,

constraint pk\_stones1 PRIMARY KEY ([Item Number]), --מפתח ראשי

constraint check\_weight1 CHECK (Weight > 0),

constraint check\_price1 CHECK (Price > 0),

)

CREATE TABLE RINGS (

[Item Number] Varchar(20) not null,

[Jewelry Type] Varchar(16) not null foreign key references [Jewelry Type] ([Jewelry Type]),

Metal Varchar(20) not null foreign key references Metal(Metal),

[Side Stones] Varchar(50) null,

Weight real not null,

Price real not null,

[Stone Number] Varchar(20) null,

constraint pk\_rings PRIMARY KEY ([Item Number]), --מפתח ראשי

constraint fk\_Stones FOREIGN KEY ([Stone Number]) REFERENCES STONES ([Item Number]), --מפתח זר

constraint check\_weight CHECK (Weight > 0),

)

CREATE TABLE Flourescence ( --LookUp table - #0.6

Flourescence varchar(6) not null primary key

--lookup table for 'Flourescence'

)

insert into Flourescence(Flourescence) values ('None'),('Faint'),('Medium'),('Strong')

CREATE TABLE DIAMONDS (

[Item Number] Varchar(20) not null, --מפתח ראשי וזר

[Diamond Measurements] Varchar(15) not null,

[Color Grade] Varchar(50) not null,

[Clarity Grade] Varchar(10) not null,

[Diamond Carat] real not null,

Polish Varchar(9) not null foreign key references Grade (grade),

Symmetry Varchar(9) not null foreign key references Grade (grade),

Flourescence Varchar(6) not null foreign key references Flourescence(Flourescence),

constraint pk\_stones2 PRIMARY KEY ([Item Number]) , --מפתח ראשי

constraint fk\_stones2 FOREIGN KEY ([Item Number]) REFERENCES STONES ([Item Number]),

constraint check\_measurements CHECK ([Diamond Measurements] LIKE ('%x% mm')),

constraint check\_carat CHECK ([Diamond Carat] BETWEEN 0 AND 30),

)

CREATE TABLE GEMSTONES (

[Item Number] Varchar(20) not null, --מפתח זר

[Gemstones Measurements] Varchar(20) not null,

Color Varchar(20) not null,

[Gemstones Carat] real not null,

Luster Varchar(9) not null foreign key references Grade (grade),

Saturation Varchar(9) not null foreign key references Grade (grade),

Brightness Varchar(9) not null foreign key references Grade (grade),

Origin Varchar(20) null,

constraint pk\_stones3 PRIMARY KEY ([Item Number]) , --מפתח ראשי

constraint fk\_stones4 FOREIGN KEY ([Item Number]) REFERENCES STONES ([Item Number]),

constraint check\_Measurements4 CHECK ([Gemstones Measurements] LIKE ('%x%x% mm')),

constraint check\_carat4 CHECK ([Gemstones Carat] BETWEEN 0 AND 100),

)

CREATE TABLE ORDERRING (

[Order ID] Varchar(20) not null,

[Ring ID] Varchar(20) not null,

Quantity int not null DEFAULT 1,

Size real not null,

constraint pk\_orderring1 PRIMARY KEY ([Order ID], [Ring ID]) , --מפתח ראשי

constraint fk\_orderring1 FOREIGN KEY ([Order ID]) REFERENCES ORDERS ([Order ID]),

constraint fk\_orderring2 FOREIGN KEY ([Ring ID]) REFERENCES RINGS ([Item Number]),

)

CREATE TABLE ORDERSTONE (

[Order ID] Varchar(20) not null,

[Stone ID] Varchar(20) not null,

Quantity int not null DEFAULT 1,

constraint pk\_orderstone1 PRIMARY KEY ([Order ID], [Stone ID]) , --מפתח ראשי

constraint fk\_orderstone1 FOREIGN KEY ([Order ID]) REFERENCES ORDERS ([Order ID]),

constraint fk\_orderstone2 FOREIGN KEY ([Stone ID]) REFERENCES STONES ([Item Number]),

)

CREATE TABLE ENGRAVING (

Characters Varchar(100) not null,

Price real not null,

constraint pk\_engraving PRIMARY KEY (Characters) , --מפתח ראשי

)