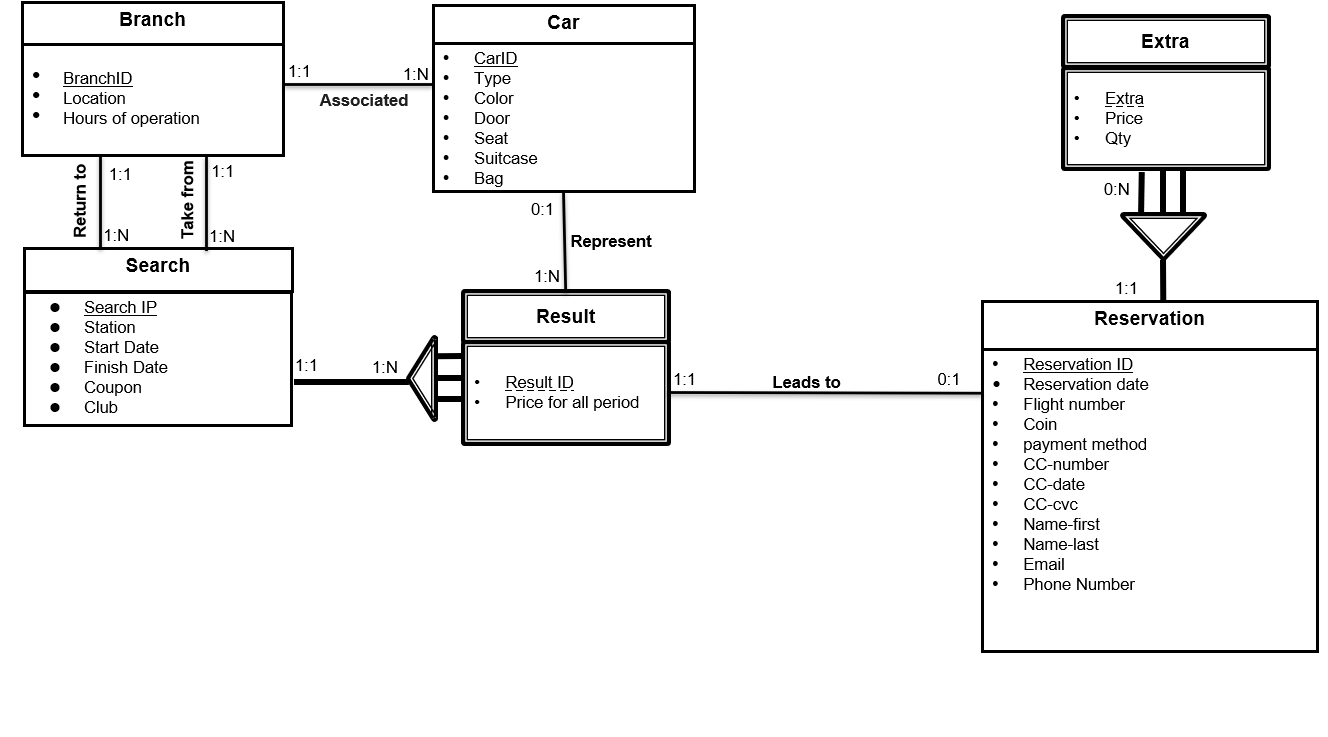
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מס' קבוצה | אתר | | | תאריך הגשה |
| 20 | [השכרת רכב בחו"ל](about:blank) | | | 26.01.24 |
| חברי הצוות - מספרי ת.ז | | | | |
| 322816885 | | 207476151 | 206364911 | |

**חלק ג'-פרק ראשון – מטלות חובה**

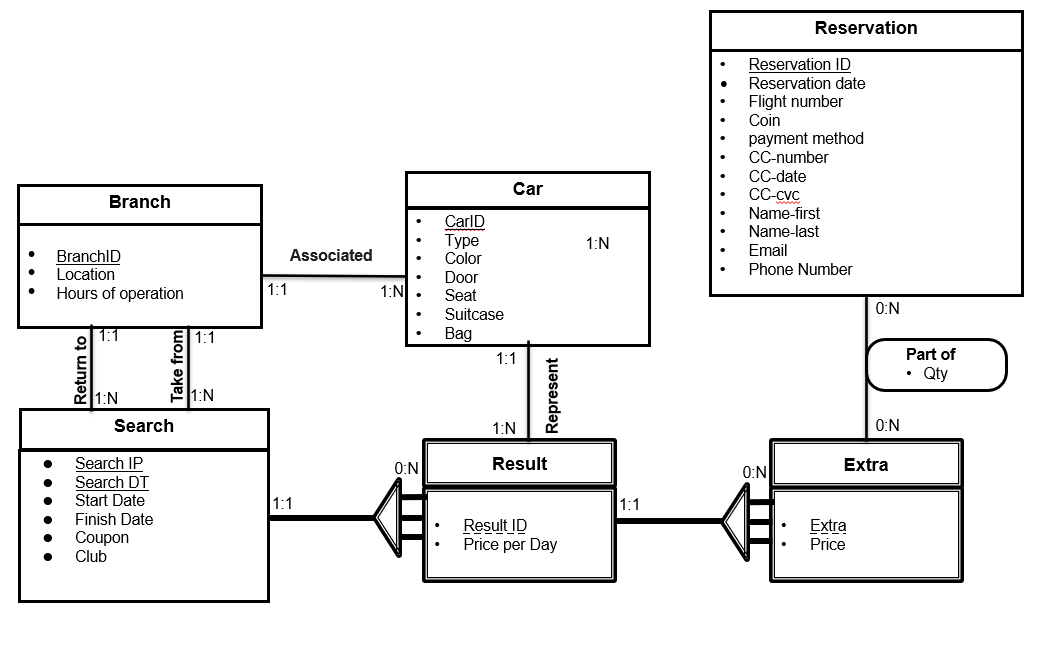
**מטלת חובה מקדימה – תיקון ה-ERD והרחבת היקף בסיס נתונים**

**ה-ERD ומבנה הטבלאות בהגשת חלק ב':**



* **BRANCHES** (BranchID, Location, HoursOfOperation)
* **CARS** (CarID, Car Type, Gear,Color, Door, Seat, Suitcase, Bag, BranchID(BRANCHES))
* **SEARCHES** (Search IP, Station, Start Date, Finish Date, Club, BranchT (BRANCHES), BranchR(BRANCHES))
* **RESULTS**(SearchID(SEARCHES), ResultID, Price for all period, CarID(CARS)))
* **CARDS**(CC-number, CC-date, CC-cvc)
* **RESERVATIONS (**ReservationID, Reservation date, Flight number, Coin, Payment-method, Name-first, Name-last, Email, Phone Number,CC-number(CARDS), CarID (CARS))  
   {ERSSearchID, ResultID}(RESULTS))
* **EXTRAS** (Reservation ID(RESERVATIONS), Extra, Price, Qty)

תרשים ה-ERD המתוקן עליהם הסתמכנו בחלק זה של העבודה:



תיאור מילולי לתיקונים:

1. מאחר ויכולים לבצע מספר חיפושים מאותו המחשב (אותה כתובת IP) הוספנו לישות Search מזהה נוסף Search DT ההופך את המזהה של הישות לייחודי, במחשב מסוים יכול להתבצע בדיוק חיפוש אחד בתאריך ושעה מסוימים.
2. הורדנו את השדה Station מהישות Search מאחר והמשמעות שלו מתבטאת בקשרים בין Search ל- Branch.
3. שינינו את הקרדינליות בין Search ל-Result (מ-0:1 ל-1:1) מהסיבה שאם ביצענו חיפוש ולא הופיעה אף מכונית - פשוט לא יהיו לי מופעים של הישות Result , ולכן תוצאה אחת או יותר מייצגת מכונית אחת בדיוק.
4. שינינו את הישות אקסטרה להיות ישות חלשה של Result כיוון ש Result מייצג לנו תוצאה של מכונית מסוימת ותוספות לרכב זמינות פר מכונית, בנוסף קישרנו את אקסטרה לישות Reservation והורדנו את השדה QTY מאקסטרה להיות תכונה של הקשר בין הישויות.

**מבנה הטבלאות המתוקן:**

● **BRANCHES** (BranchID, Location, HoursOfOperation)

● **CARS** (CarID, Gear, Color, BranchID(BRANCHES), Car Type(CAR TYPES))

● **CAR TYPES** (Car Type, Door, Seat, Suitcase, Bag)

● **SEARCHES** (Search IP, Search DT, Start Date, Finish Date, Club, BranchT (BRANCHES), BranchR(BRANCHES))

● **RESULTS**({SearchIP,SearchDT} (SEARCHES), ResultID, Price per day , CarID(CARS)))

● **CARDS**(CC-number, CC-date, CC-cvc)

● **EXTRAS** ({SearchID,SearchDT,ResultID}(RESULTS), Extra, Price)

● **RESERVATIONS** (ReservationID, Reservation date, Flight number, Coin, Payment-method, Name-first, Name-last, Email, Phone Number, CC-number(CARDS))

* **EXTRASOFRESERVATION** ({SearchID,SearchDT,ResultID,Extra}(EXTRAS), ReservationID)(RESERVATIONS) , Qty)

**הנחות נוספות:**

1. בטבלה reservations בשדה club הערך הוא 0 עבור לקוחות במחלקת Private’', ו-1 עבור'Business'.

2. השדה Extra והשדה Price בישות Extra יכולים לקבל ערכים של ‘NO EXTRA’ ו-0 בהתאמה.

**מטלה 1 (35%) – שאילתות**

**שתי שאילתות SELECT ללא קינון (5%, 2.5% לכל שאילתה)**

* + - 1. הצג את פרטי החיפוש של חיפושים אשר לא הובילו להזמנה ? יש להציג בסדר יורד לפי זמן החיפוש
* הגיון עסקי:

**ברצון החברה להשתפר ולהתפתח בצורה שתוביל לעלייה במכירות שלה, על מנת כך עליה לחקור בהמשך את הסיבות שבגללן היו חיפושים שלא הובילו להזמנות בהמשך, האם תוצאות החיפוש לא התאימו לצרכיהם או בכלל לא היו קימות, האם המחיר היה גבוה מידיי?, האם היו חסרות תוספות מסוימות לרכב ? על מנת לענות על השאלה שלב מקדים הוא לקבץ את החיפושים הללו.**

SELECT S.SearchDT,S.SearchIP,S.StartDate,S.FinishDate,S.BranchT,S.BranchR,R.ResultID, E.Extra

FROM SEARCHES AS S LEFT JOIN RESULTS AS R ON S.SearchIP = R.SearchIP

AND S.SearchDT = R.SearchDT LEFT JOIN

EXTRAS AS E ON R.SearchIP = E.SearchIP

AND R.SearchDT = E.SearchDT

AND R.ResultID = E.ResultID LEFT JOIN

EXTRASOFRESERVATION AS ER ON E.SearchIP = ER.SearchIP

AND E.SearchDT = ER.SearchDT

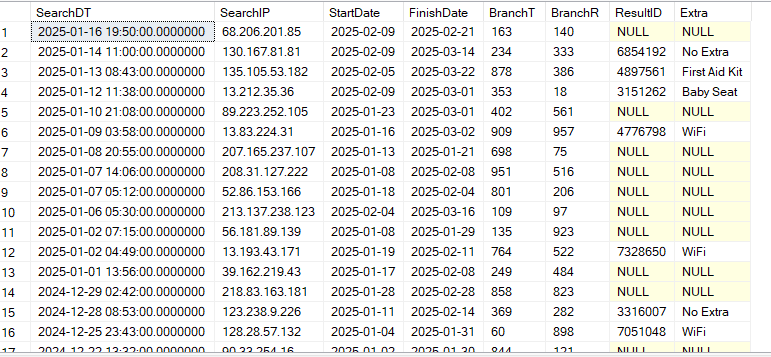
AND E.ResultID = ER.ResultID

AND E.Extra = ER.Extra

WHERE ER.ReservationID IS NULL  
ORDER BY S.SearchDT DESC

* פלט השאילתה:

1026 רשומות



* + - 1. שאילתה עם מרכיבי GROUP BY ו- HAVING ופונקציית המערכת:  
         מהי כמות ההזמנות הכוללת בסדר יורד עבור כל סוג רכב המוחזק באוויס שמחיר השכרתו עולה על 1000 וכמות הזמנות עולה על 2 ?
* הגיון עסקי:

**החברה רוצה לדעת איזה רכבים מניבים להם הכנסה גבוה יחסית ובכך לשער איזה רכבים כדאי לה לקנות.**

SELECT C.CarType,R.PricePerDay,

TOTAL\_RENT=COUNT(RS.RESERVATIONID)

FROM CARS AS C JOIN RESULTS AS R ON C.CARID=R.CARID

JOIN EXTRASOFRESERVATION AS E ON E.RESULTID=R.RESULTID

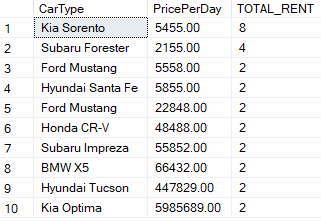
JOIN RESERVATIONS AS RS ON RS.RESERVATIONID=E.RESERVATIONID

GROUP BY C.CARTYPE,R.PricePerDay

HAVING COUNT(RS.RESERVATIONID) >= 2 AND R.PricePerDay > 1000

ORDER BY TOTAL\_RENT DESC

* פלט השאילתה:



**שתי שאילתות SELECT מקוננות (10%, 5% לכל שאילתה)**

**שאילתה מקוננת המקבלת ערך סקלרי (קינון ב-having):**הצג בסדר יורד את הסניפים שממוצע המכירות שלהם נמוך מממוצע המכירות הכללי ואת ממוצע המכירות

* הגיון עסקי:

השאילתה מזהה סניפים שהביצועים שלהם (במונחי ממוצע הכנסות ממכירות) נמוכים מהממוצע הכולל של כל הסניפים. כך ניתן להתמקד בסניפים החלשים כדי לשפר את ביצועיהם.

select c.branchID,

year\_avg\_income=avg(RE.pricePerDay\* DATEDIFF(DAY,S.StartDate,S.FinishDate)+EXR.Qty\*E.Price)

from reservations as R join extrasofreservation

as EXR on EXR.reservationID=R.reservationID

join RESULTS AS RE ON RE.ResultID=EXR.ResultID

join cars as c on c.CarID=re.CarID

join searches as S on S.SearchIP=RE.SearchIP AND S.SearchDT=RE.SearchDT

join EXTRAS as E on E.SearchIP=S.SearchIP AND S.SearchDT=E.SearchDT AND E.ResultID = RE.ResultID

group by C.BranchID

having avg(RE.pricePerDay\* DATEDIFF(DAY,S.StartDate,S.FinishDate)+EXR.Qty\*E.Price)<

(

select avg(RE.pricePerDay\* DATEDIFF(DAY,S.StartDate,S.FinishDate)+EXR.Qty\*E.Price)

from reservations as R join extrasofreservation

as EXR on EXR.reservationID=R.reservationID

join RESULTS AS RE ON RE.ResultID=EXR.ResultID

join cars as c on c.CarID=re.CarID

join searches as S on S.SearchIP=RE.SearchIP AND S.SearchDT=RE.SearchDT

join EXTRAS as E on E.SearchIP=S.SearchIP and S.SearchDT=E.SearchDT AND E.ResultID = RE.ResultID

)

order by year\_avg\_income desc

* פלט השאילתה:

667 רשומות

**A screenshot of a table with numbers

Description automatically generated**

**שאילתה מקוננת המקבלת גם ערך סקלרי וגם טבלאי (קינון ב-having וב-from** ):  
הצג את כל התוספות הנמכרות ביותר על פי סוג מכונית.

* הגיון עסקי:

מטרת השאילתה היא **לזהות את התוספות הפופולריות ביותר לכל סוג רכב**, על מנת למקסם רווחים, לייעל את ניהול המלאי, לשפר את אסטרטגיית השיווק, ולהתאים את חוויית הלקוח לצרכים המדויקים שלו.

SELECT CarType, COUNT\_EXTRA.Extra, MaxQty= MAX(COUNT\_EXTRA.CountExtra)

FROM (

SELECT

C.CarType,

ER.Extra,

CountExtra=SUM(ER.Qty)

FROM CARS AS C

JOIN RESULTS AS R ON C.CarID = R.CarID

JOIN EXTRAS AS E ON R.ResultID = E.ResultID

JOIN EXTRASOFRESERVATION AS ER ON

E.SearchIP = ER.SearchIP AND

E.SearchDT = ER.SearchDT AND

E.ResultID = ER.ResultID AND

E.Extra = ER.Extra

GROUP BY C.CarType, ER.Extra

) AS COUNT\_EXTRA

WHERE COUNT\_EXTRA.CountExtra = (

SELECT MAX(Max\_Extra.CountExtra)

FROM (

SELECT

C2.CarType,

ER2.Extra,

CountExtra=SUM(ER2.Qty)

FROM CARS AS C2

JOIN RESULTS AS R2 ON C2.CarID = R2.CarID

JOIN EXTRAS AS E2 ON R2.ResultID = E2.ResultID

JOIN EXTRASOFRESERVATION AS ER2 ON

E2.SearchIP = ER2.SearchIP AND

E2.SearchDT = ER2.SearchDT AND

E2.ResultID = ER2.ResultID AND

E2.Extra = ER2.Extra

GROUP BY C2.CarType, ER2.Extra

) AS Max\_Extra

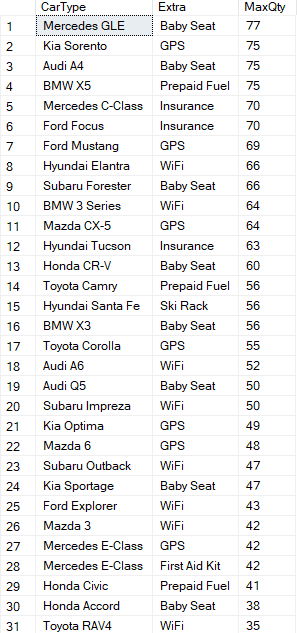
WHERE Max\_Extra.CarType = COUNT\_EXTRA.CarType

)

GROUP BY COUNT\_EXTRA.CarType, COUNT\_EXTRA.Extra,COUNT\_EXTRA.CountExtra   
order by COUNT\_EXTRA.CountExtra desc

* פלט השאילתה:

31 רשומות

****

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

1. **שאילתה המשלבת פונקציית חלון(שימוש ב** ()**sum(), rank):**

לכל סניף ולכל שנת פעילות מה סכום המכירות הכולל עבור אותה שנה, מה סכום המכירות המצטבר לאורך השנים עבור הסניף, ומה המדרג של השנה לעומת השנים האחרות בהתייחס למכירות שבוצעו על ידי הסניף. ההיגיון העסקי מאחורי שאילתה זו היא לבחון איזה סניפים הינם הרווחים ביותר, לראות היכן המכירות ירדו ולהבין את הסיבה לירידה.

SELECT

C.BranchID,

Year\_Of\_Reservation=YEAR(RES.ReservationDate) ,

Yearly\_Amount=SUM(R.PricePerDay \* DATEDIFF(DAY, S.StartDate, S.FinishDate) + EXR.Qty \* E.Price),

Cumulative\_Amount=SUM(SUM(R.PricePerDay \* DATEDIFF(DAY, S.StartDate, S.FinishDate) + EXR.Qty \* E.Price))

OVER (PARTITION BY C.BranchID ORDER BY YEAR(RES.ReservationDate)),

Year\_Rank=RANK() OVER (PARTITION BY C.BranchID ORDER BY SUM(R.PricePerDay \* DATEDIFF(DAY, S.StartDate, S.FinishDate) + EXR.Qty \* E.Price) DESC)

FROM

SEARCHES AS S

JOIN

RESULTS AS R

ON S.SearchIP = R.SearchIP AND S.SearchDT = R.SearchDT

JOIN

CARS AS C

ON C.CarID = R.CarID

JOIN

EXTRAS AS E

ON E.SearchIP = S.SearchIP AND E.SearchDT = S.SearchDT

JOIN

EXTRASOFRESERVATION AS EXR

ON E.ResultID = EXR.ResultID AND E.Extra = EXR.Extra

AND E.SearchIP = EXR.SearchIP AND E.SearchDT = EXR.SearchDT

JOIN

RESERVATIONS AS RES

ON EXR.ReservationID = RES.ReservationID

GROUP BY

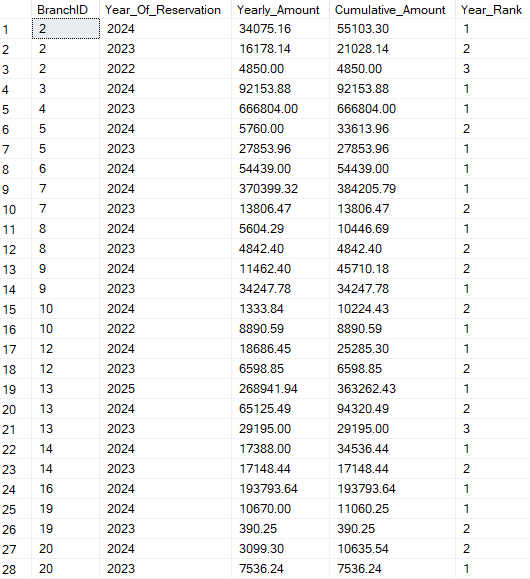
C.BranchID, YEAR(RES.ReservationDate)

ORDER BY

Year\_Of\_Reservation DESC;

* פלט השאילתה:

1134 רשומות



1. **שאילתה המשלבת פונקציית חלון שימוש ב LAG() וROW NUMBER():**

**עבור השכרות של סוג רכב מסוים בסניף מסוים מה המספר הסידורי של הרכב לפי המחיר הגבוהה לנמוך, בנוסף עבור כל סוג רכב בסניף הצג מה היה מחירו בהשכרה הקודמת.**

* הגיון עסקי:  
  המטרה העסקית של השאילתה היא לעקוב אחרי שינויים במחיר השכרת רכבים מאותו הסוג בסניף בו מבוצעת ההשכרה, ולהשוות את המחיר הנוכחי של רכב למחיר שהיה בהשכרה הקודמת. כך ניתן לזהות מגמות במחירים (האם המחיר עלה או ירד), ולהבין אם יש צורך לבצע התאמות במחירים על פי נתוני השוק. בנוסף השאילתה מדרגת את ההשכרה לפי מכיר השכרת הרכב ליום בסדר עולה מהמחיר הגבוה לנמוך.

SELECT

B.BranchID, R.PricePerDay,C.CarType,

S.startdate,

pricegroup=ROW\_NUMBER() OVER (PARTITION BY B.BranchID ORDER BY R.PricePerDay desc),

prev\_price=LAG(R.PricePerDay) OVER (PARTITION BY B.BranchID,C.CarType ORDER BY S.startdate asc)

FROM

RESULTS AS R JOIN CARS AS C ON R.CarID = C.CarID JOIN

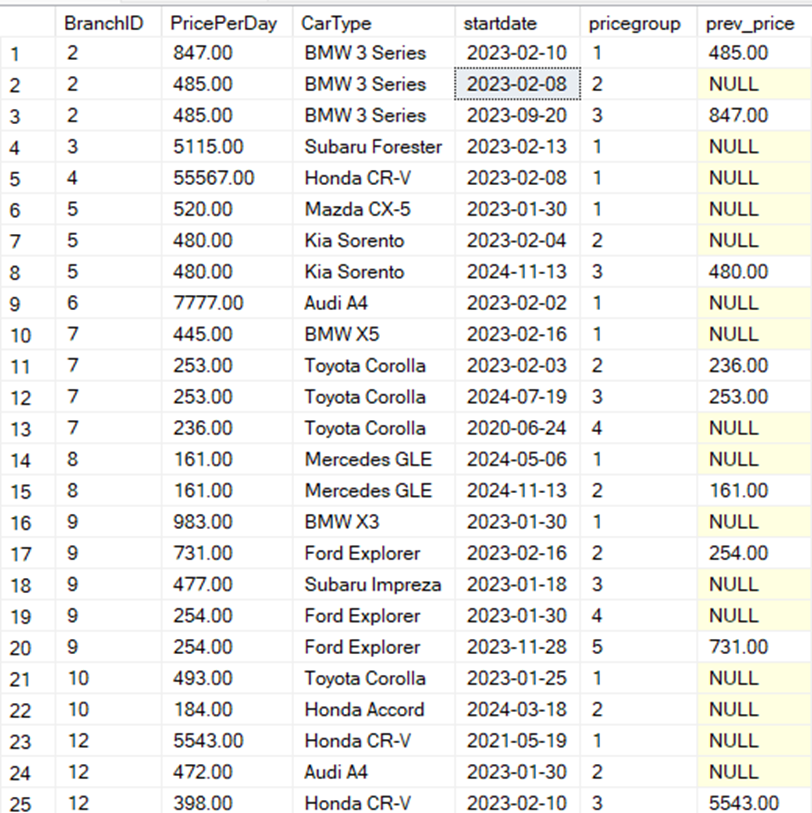
SEARCHES AS S ON R.SearchIP = S.SearchIP AND R.SearchDT = S.SearchDT

JOIN BRANCHES AS B ON C.BranchID = B.BranchID

ORDER BY

B.BranchID, pricegroup;

* פלט השאילתה: 2125 רשומות

****

**שאילתה מקוננת תוך שימוש ב-CTE (פסקת WITH) (10%)**

השאילתה תתמקד בחישוב פרטים הקשורים לחיפושים, מכירות והפסדים אפשריים.

1. מספר החיפושים שלא הובילו להזמנה.
2. חישוב משך השכרת רכב ממוצע עבור חיפושים שהובילו להזמנה.
3. חישוב ממוצע ההכנסות עבור יום השכרה, על סמך ההזמנות שבוצעו.
4. הפקת דו"ח הכולל הערכת הפסדים פוטנציאליים עבור חיפושים שלא הובילו להזמנה.

* הגיון עסקי: **חיפושים שלא הובילו להזמנה חושפים חולשות בתהליך המכירה, משך ההשכרה הממוצע מגלה מגמות שימוש ממוצעות של לקוחות אוויס, חישוב ההפסדים הפוטנציאליים הוא כלי חשוב שיכול לתרום להנעת העובדים לקידום מכירות החברה ויכול לשמש חזון להצבת יעדים לחברה בהמשך.**

With

dismissedSearches as (---חישוב מספר החיפושים לא הובילו להזמנה

select

dissmissed\_num=count (\*)

from SEARCHES as S left join RESULTS as R on S.SearchDT=R.SearchDT and S.searchIP=R.SearchIP

left join EXTRAS as E on E.SearchIP=R.SearchIP and E.SearchDT=R.SearchDT and E.ResultID=R.ResultID

left join EXTRASOFRESERVATION as EX on E.SearchIP=EX.SearchIP and E.SearchDT=EX.SearchDT and E.ResultID=EX.ResultID and E.Extra=EX.Extra

left join RESERVATIONS as RE on RE.ReservationID=EX.ReservationID

where RE.ReservationID is null

),

days\_rent as (---מספר ימי ההזמנות הממוצע להשכרת רכב

select

Reservation\_Num=count (\*),

Avg\_Rent\_Dayes=avg(datediff(dd,S.startDate,S.FinishDate))

from SEARCHES as S join RESULTS as R on S.SearchDT=R.SearchDT and S.searchIP=R.SearchIP

join EXTRAS as E on E.SearchIP=R.SearchIP and E.SearchDT=R.SearchDT and E.ResultID=R.ResultID

join EXTRASOFRESERVATION as EX on E.SearchIP=EX.SearchIP and E.SearchDT=EX.SearchDT and E.ResultID=EX.ResultID and E.Extra=EX.Extra

join RESERVATIONS as RE on RE.ReservationID=EX.ReservationID

),

Avg\_Dismissed\_Income as (---חישוב ממוצע ההכנסות עבור יום השכרה

select

Avg\_Income\_Of\_Rental\_Day=round(AVG(R.pricePerDay),2)

from SEARCHES as S join RESULTS as R on S.SearchDT=R.SearchDT and S.searchIP=R.SearchIP

join EXTRAS as E on E.SearchIP=R.SearchIP and E.SearchDT=R.SearchDT and E.ResultID=R.ResultID

join EXTRASOFRESERVATION as EX on E.SearchIP=EX.SearchIP and E.SearchDT=EX.SearchDT and E.ResultID=EX.ResultID and E.Extra=EX.Extra

join RESERVATIONS as RE on RE.ReservationID=EX.ReservationID

)

select DS.dissmissed\_num,DR.Avg\_Rent\_Dayes,AI.Avg\_Income\_Of\_Rental\_Day,

Avg\_Dismissed\_Income=round(DS.dissmissed\_num\*DR.Avg\_Rent\_Dayes\*AI.Avg\_Income\_Of\_Rental\_Day,2)

from dismissedSearches as DS join days\_rent as DR on 1=1

join Avg\_Dismissed\_Income as AI on 1=1;

* פלט השאילתה:



## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

# מטלה 2 (35%) – יישומי כלים מתקדמים

# View (5%)

### VIEW עבור סיכום ההכנסות השנתיות לסניפים: נניח שברצון המנכ"ל לחשב את ההכנסות השנתיות לכל סניף, כולל ההזמנות שבוצעו והתוספות שהוזמנו. זה יכול לעזור למנהלי סניפים להבין את ביצועיהם ביחס לשנים.

**מורכבות השאילתה:**  
ה-VIEW מחלץ נתונים ממספר טבלאות (RESERVATIONS, EXTRASOFRESERVATION, RESULTS BRANCHES, SEARCHES) ומשתמש בפונקציות כמו SUM, DATEDIFF .   
בכך הוא עונה על הקריטריון של שאילתה מורכבת שמבוצעת לעיתים קרובות.

CREATE VIEW vw\_Yearly\_Revenue\_Per\_Branch AS

SELECT

B.BranchID,

B.Location,

Year\_Of\_Reservation= YEAR(R.ReservationDate) ,

Total\_Revenue=SUM(RE.PricePerDay \* DATEDIFF(DAY, S.StartDate, S.FinishDate) + EXR.Qty \* E.Price)

FROM

reservations as R left join extrasofreservation

as EXR on EXR.reservationID=R.reservationID

join RESULTS AS RE ON RE.ResultID=EXR.ResultID

join cars as c on c.CarID=re.CarID

join searches as S on S.SearchIP=RE.SearchIP

left join EXTRAS as E on E.SearchIP=S.SearchIP

join BRANCHES as B on B.BranchID=C.BranchID

GROUP BY

B.BranchID, B.Location, YEAR(R.ReservationDate);

# פונקציות (Functions) (10%, 2 פונקציות – 5% לכל פונקציה)

Tabular Function -פונקציה המחזירה טבלה:

**הפונקציה תחשב את ממוצע ההכנסות השנתי לכל סניף לפי שנה מסוימת, ותדרג את ממוצע המחירות של הסניף לאורך השנים על בסיס הממוצע.**

הגיון עסקי:

**ההיגיון העסקי מאחורי הפונקציה הוא שיפור ניהול והבנת ביצועי הסניפים בעסק. הפונקציה מאפשרת להעריך ולהשוות את הביצועים של סניפים שונים לפי הכנסותיהם השנתיות, ובכך להקל על קבלת החלטות עסקיות.**

create function Get\_Branch\_Yearly\_Sales\_rank\_VW (@BranchID int)

returns table

as

return

(select BranchID,

avg\_yearly\_sales=avg(Total\_Revenue),

from vw\_Yearly\_Revenue\_Per\_Branch

where (BranchID=@BranchID)

group by BranchID

)

דוגמא לשימוש בפונקציה:

SELECT \*

FROM dbo.Get\_Branch\_Yearly\_Sales\_Rank\_VW (9)

פלט השאילתה:



Scalar Function -פונקציה המחזירה סקלר:

הפונקציה CarAvailability נועדה לשפר את ניהול צי הרכבים של החברה ולספק מידע מדויק על הזמינות של רכבים בסניפים בזמן אמת**.**

הגיון עסקי:

מתן מידע בזמן אמת על מספר הרכבים הזמינים. זה מאפשר למנהלי סניפים לנהל את הצי בצורה אופטימלית, למנוע טעויות בהזמנות, ולהבטיח שיפור בשירות הלקוחות.

CREATE FUNCTION CarAvailability (@BRANCHID INT, @CHEAKDATE DATE )

RETURNS int

AS BEGIN

DECLARE @BRANCHCARS Int

SELECT @BRANCHCARS = COUNT (CARS.BranchID)

FROM CARS

WHERE CARS.BranchID = @BRANCHID

DECLARE @RentedCars INT;

SELECT @RentedCars = COUNT(\*)

FROM SEARCHES AS S JOIN RESULTS AS R ON

R.SearchIP=S.SearchIP

JOIN EXTRAS AS E ON

E.ResultID=R.ResultID

JOIN CARS AS C ON

C.CarID=R.CarID

JOIN EXTRASOFRESERVATION AS ER ON

ER.ResultID=E.ResultID

JOIN RESERVATIONS AS RES ON

ER.ReservationID=RES.ReservationID

WHERE C.BranchID = @BranchID

AND @CHEAKDATE BETWEEN S.StartDate AND S.FinishDate

SET @BRANCHCARS=@BRANCHCARS-@RentedCars ;

RETURN @BRANCHCARS  
 END

דוגמה לשימוש בפונקציה:

SELECT

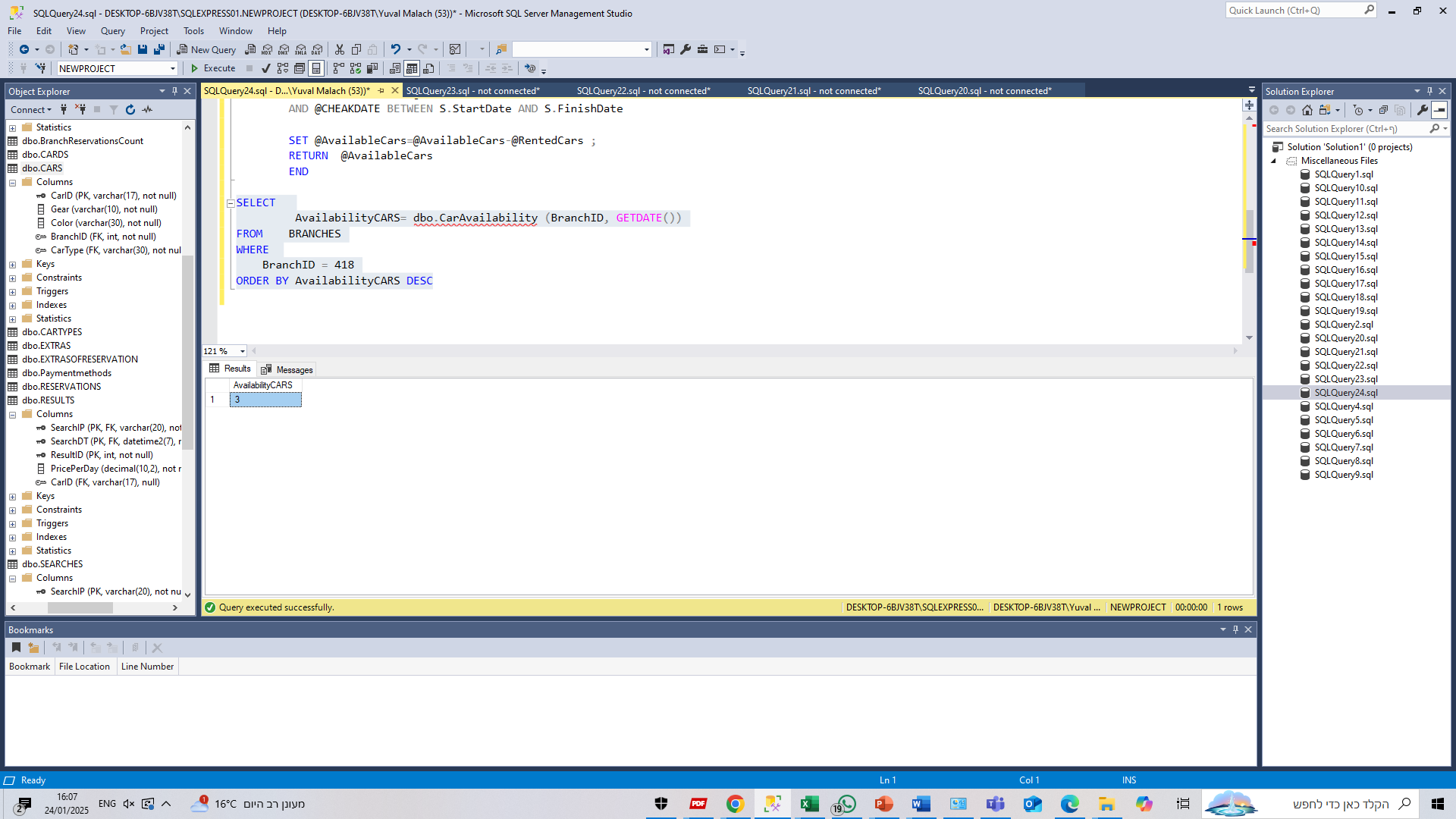
AvailabilityCARS= dbo.CarAvailability (BranchID, GETDATE())

FROM BRANCHES

WHERE

BranchID = 418

פלט השאילתה:



# Trigger (10%)

הטריגר מופעל בעקבות הוספה של מכונית לסניף מסוים או הורדת מכונית בשל סיבה מסוימת. המטרה הינה לדעת כמה מכוניות שייכות לכל סניף.   
הסיבה שאנו רוצים לדעת כמה מכוניות יש בכל סניף היא בשביל לדעת אם יש מחסור במכוניות בסניף מסוים או להפך יש יותר מדי מכוניות ואין ביקוש למכוניות וניתן להעביר מכוניות לסניף אחר.   
בשביל להמחיש את הטריגר הוספנו שדה מחושב בטבלת סניפים אשר סופר כמה מכוניות יש בכל סניף.

ALTER TABLE BRANCHES

ADD AvailableCars INT NOT NULL DEFAULT 0;

UPDATE BRANCHES

SET AvailableCars = (

SELECT COUNT(\*)

FROM CARS

WHERE CARS.BranchID = BRANCHES.BranchID

)

--DROP TRIGGER IF EXISTS AfterCarInsert

CREATE TRIGGER AfterCarInsert

ON CARS for INSERT AS UPDATE BRANCHES

SET AvailableCars =( ISNULL(AvailableCars,0)+(

SELECT COUNT(\*)

FROM INSERTED as I

WHERE I.BRANCHID = BRANCHES.BranchID)

)

--DROP TRIGGER IF EXISTS AfterCarDelete

CREATE TRIGGER AfterCarDelete

ON CARS for DELETE AS UPDATE BRANCHES

SET AvailableCars =( ISNULL(AvailableCars,0)-(

SELECT COUNT(\*)

FROM deleted as D

WHERE D.BRANCHID = BRANCHES.BranchID))

מימוש הטריגר:

INSERT INTO CARS (CarID, Gear, Color, BranchID, CarType)

VALUES ('444', 'Automatic', 'Red', 2, 'Kia Sorento'),

('781', 'Automatic', 'Red', 2, 'Kia Sorento'),

('54189', 'Automatic', 'Red', 2, 'Kia Sorento'),

('88997', 'Automatic', 'Red', 2, 'Kia Sorento');

DELETE FROM CARS WHERE CarID = '781'

DELETE FROM CARS WHERE CarID = '54189'

DELETE FROM CARS WHERE CarID = '444';

SELECT \* FROM BRANCHES WHERE BranchID = 2

לפני הטריגר:

תמונה שמכילה טקסט, צילום מסך, תצוגה, תוכנה

התיאור נוצר באופן אוטומטי

אחרי הטריגר:

תמונה שמכילה טקסט, צילום מסך, תצוגה, תוכנה

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, תוכנה, תצוגה

התיאור נוצר באופן אוטומטי

# פרוצדורה שמורה ((Stored Procedure (10%)

**הגיון עסקי:**  
פרוצדורה זו עוזרת לזהות את הלקוחות שמניבים את הרווח הגבוה ביותר לעסק. זיהוי זה מאפשר למקד מאמצי שיווק ושימור לקוחות בלקוחות המובילים. בנוסף, בזכות מידע זה ניתן לספק תובנות לבניית תוכניות נאמנות מותאמות אישית כגון הנחות או מבצעים ללקוחות שמוציאים סכומים גבוהים וכך מחזקת את הקשר עם לקוחות אלו ומונעת מעבר שלהם למתחרים.

DROP PROCEDURE GetTopCustomers

CREATE PROCEDURE GetTopCustomers

@TopCount INT

AS

BEGIN

SELECT TOP (@TopCount)

R.Email,

TotalSpent=SUM(RT.PricePerDay \* ER.Qty + E.Price \* ER.Qty)

FROM RESERVATIONS AS R

JOIN EXTRASOFRESERVATION ER ON R.ReservationID = ER.ReservationID

JOIN RESULTS AS RT ON ER.ResultID = RT.ResultID

JOIN EXTRAS AS E ON ER.ResultID = E.ResultID

GROUP BY R.Email

ORDER BY SUM(RT.PricePerDay \* ER.Qty + E.Price \* ER.Qty) DESC;

END;

**מימוש הפרוצדורה:**

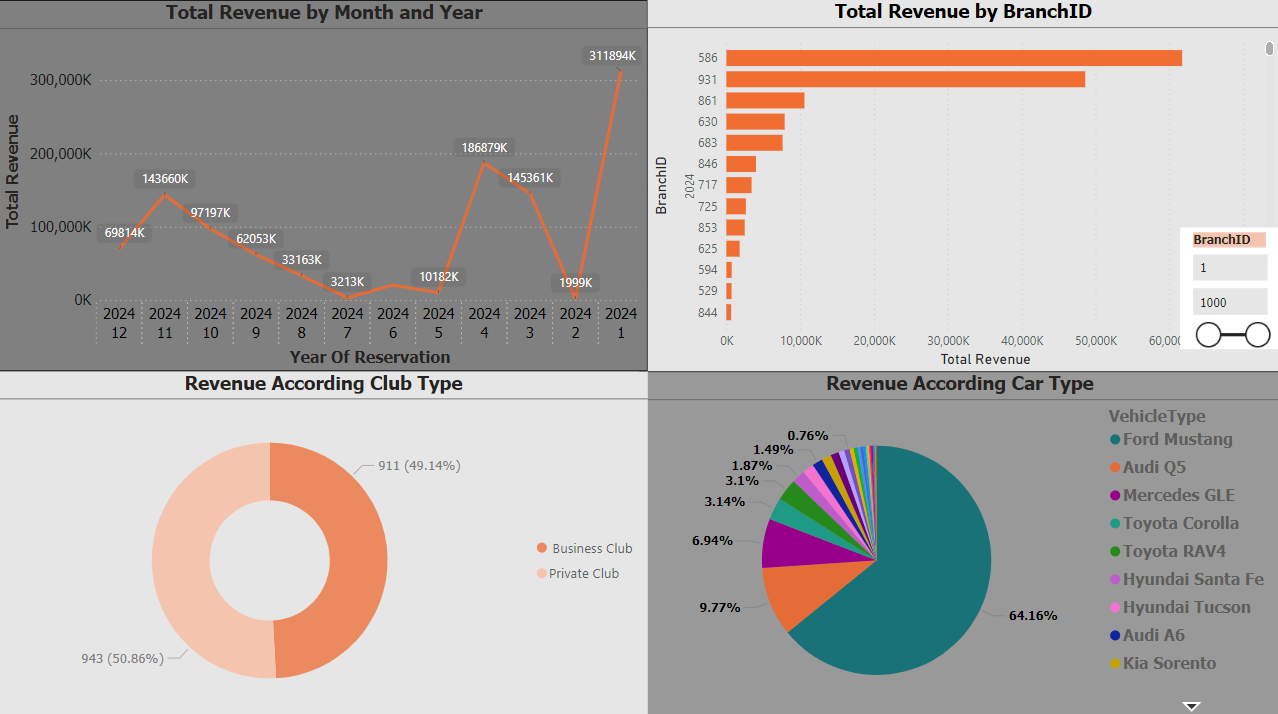
EXEC GetTopCustomers @TopCount = 5;

פלט השאילתה:

**A screenshot of a computer

Description automatically generated**

# מטלה 3 (20%) – כלים להצגת נתונים

**דוח עסקי לסמנכ"ל התפעול סמנכ"ל הכספים, מנהלי שיווק וצוותי תפעול:**

* **בקרה על הכנסות:**

הצגת סך ההכנסות לפי חודשים(“Total Revenues 2024”) מאפשרת לעקוב אחר ביצועי החברה לאורך השנה ולזהות עונתיות או מגמות בירידה/עלייה בהכנסות.

* **ניתוח סניפים:**

גרף "Total Revenue by Branches" מציג את ההכנסות הגבוהות ביותר לפי סניף, ומספק תובנות על סניפים מצליחים או כאלה שדורשים תשומת לב לשיפור.

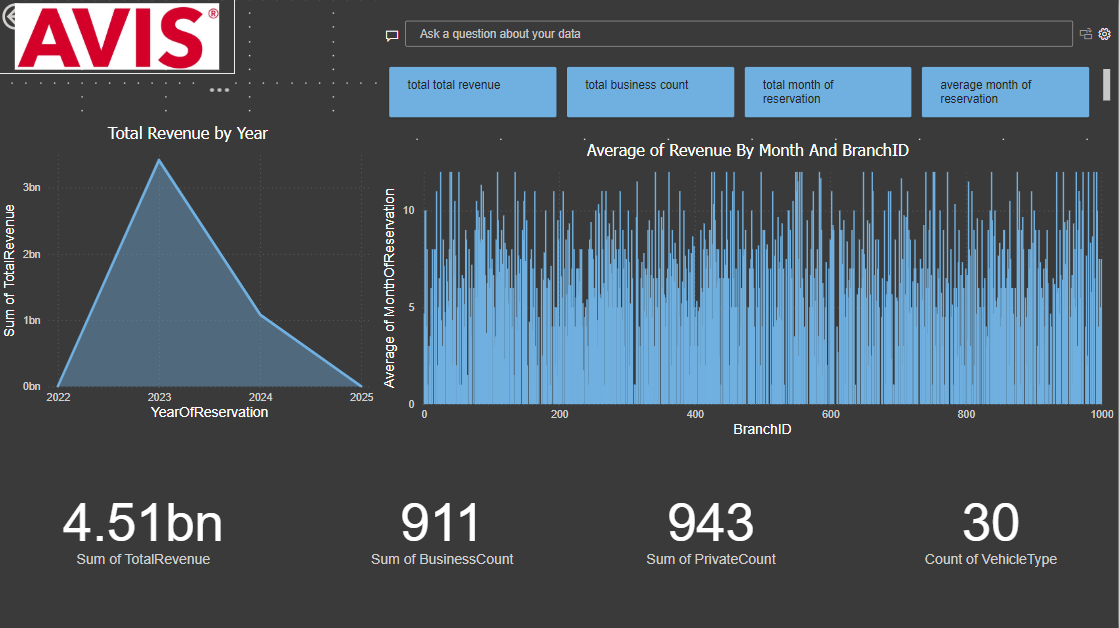
* **העדפות לקוחות לפי סוג רכב:**

הפאי-צ'ארט "Revenue according Car Type" מאפשר להבין אילו סוגי רכבים מניבים את ההכנסות הגבוהות ביותר, וכך לספק המלצות על מיקוד במלאי או במבצעי שיווק.

* **פילוח לקוחות לפי מועדון:**

גרף "Total Reservations by Customer Club Type" עוזר להבחין בין לקוחות עסקיים לפרטיים, כדי לזהות את משקל כל קבוצה ולטייב שירותים והצעות ייעודיות.

**לוח מחוונים למנכ"ל:**

****

* **גרף סך ההכנסות לפי שנים (Total Revenue by Year)**:  
  הגרף בצד שמאל למעלה מציג את סך ההכנסות לפי שנת ההזמנה. ניתן לראות מגמה של גידול משמעותי בהכנסות בשנת 2023 ולאחר מכן ירידה חדה ב-2024 ו-2025.
* **גרף ממוצע הכנסות לפי חודש וסניף :**  
  הגרף בצד ימין למעלה מציג את ממוצע ההכנסות החודשי לפי מזהה הסניף (BranchID). ככל הנראה, הוא משמש להשוואה בין סניפים שונים על בסיס הכנסות חודשיות.
* **כרטיסי מידע בתחתית**:

**4.51bn (סך ההכנסות):** מציג את סך ההכנסות הכולל.

**911 (סך ההזמנות העסקיות):** מציג את כמות ההזמנות שבוצעו על ידי לקוחות עסקיים.

**943 (סך ההזמנות הפרטיות ):** מציג את כמות ההזמנות שבוצעו על ידי לקוחות פרטיים.

**30 (סוגי רכבים):** מציג את מספר סוגי הרכבים הקיימים בהיצע של אוויס.

**קוד עזר להפקת הדוחות:**

REATE OR ALTER VIEW ReservationRevenue AS

SELECT

B.BranchID,

B.Location AS BranchLocation,

YearOfReservation = YEAR(RES.ReservationDate),

MonthOfReservation = MONTH(RES.ReservationDate),

VehicleType = CT.CarType,

TotalRevenue = SUM(RE.pricePerDay \* DATEDIFF(DAY, S.StartDate, S.FinishDate)) + SUM(EXR.Qty \* E.Price),

TotalReservations = COUNT(DISTINCT RES.ReservationID),

BusinessCount = COUNT(CASE WHEN S.Club = 1 THEN 1 END),

PrivateCount = COUNT(CASE WHEN S.Club = 0 THEN 1 END),

CustomerType = (CASE WHEN S.Club = 1 THEN 'Business' WHEN S.Club = 0 THEN 'Private'ELSE 'Unknown'END)

FROM

SEARCHES S

JOIN

RESULTS RE

ON S.SearchIP = RE.SearchIP AND S.SearchDT = RE.SearchDT

JOIN

BRANCHES B

ON S.BranchT = B.BranchID

JOIN

CARS C

ON RE.CarID = C.CarID

JOIN

CARTYPES CT

ON C.CarType = CT.CarType

JOIN

EXTRASOFRESERVATION EXR

ON RE.ResultID = EXR.ResultID

AND RE.SearchIP = EXR.SearchIP

AND RE.SearchDT = EXR.SearchDT

JOIN

EXTRAS E

ON EXR.Extra = E.Extra

AND EXR.ResultID = E.ResultID

AND EXR.SearchIP = E.SearchIP

AND EXR.SearchDT = E.SearchDT

JOIN

RESERVATIONS RES

ON RES.ReservationID = EXR.ReservationID

GROUP BY

B.BranchID,

B.Location,

CT.CarType,

YEAR(RES.ReservationDate),

S.Club,

MONTH(RES.ReservationDate);

## מטלה 4 (10%) – אופטימיזציה של שאילתות באמצעות Generative AI

* השאילתה הראשונה שבחרנו לעשות לה אופטימיזציה הינה השאילתה הבאה :

select c.branchID,

year\_avg\_income=avg(RE.pricePerDay\* DATEDIFF(DAY,S.StartDate,S.FinishDate)+EXR.Qty\*E.Price)

from reservations as R left join extrasofreservation

as EXR on EXR.reservationID=R.reservationID

join RESULTS AS RE ON RE.ResultID=EXR.ResultID

join cars as c on c.CarID=re.CarID

join searches as S on S.SearchIP=RE.SearchIP AND S.SearchDT=RE.SearchDT

join EXTRAS as E on E.SearchIP=S.SearchIP AND S.SearchDT=E.SearchDT AND E.ResultID = RE.ResultID

group by C.BranchID

having avg(RE.pricePerDay\* DATEDIFF(DAY,S.StartDate,S.FinishDate)+EXR.Qty\*E.Price)<

(

select avg(RE.pricePerDay\* DATEDIFF(DAY,S.StartDate,S.FinishDate)+EXR.Qty\*E.Price)

from reservations as R join extrasofreservation

as EXR on EXR.reservationID=R.reservationID

join RESULTS AS RE ON RE.ResultID=EXR.ResultID

join cars as c on c.CarID=re.CarID

join searches as S on S.SearchIP=RE.SearchIP AND S.SearchDT=RE.SearchDT

join EXTRAS as E on E.SearchIP=S.SearchIP and S.SearchDT=E.SearchDT AND E.ResultID = RE.ResultID

)

order by year\_avg\_income desc

**השאילתה אשר הבינה המלאכותית בנתה:**

CREATE INDEX idx\_reservations\_reservationid ON RESERVATIONS(ReservationID);

CREATE INDEX idx\_extrasofreservation ON EXTRASOFRESERVATION(ReservationID, ResultID, Qty);

CREATE INDEX idx\_results ON RESULTS(ResultID, SearchIP, SearchDT, pricePerDay, CarID);

CREATE INDEX idx\_cars ON CARS(CarID, BranchID);

CREATE INDEX idx\_searches ON SEARCHES(SearchIP, SearchDT);

CREATE INDEX idx\_extras ON EXTRAS(SearchIP, SearchDT, ResultID, Price);

SELECT

C.BranchID,

AVG(RE.pricePerDay \* DATEDIFF(DAY, S.StartDate, S.FinishDate) + EXR.Qty \* E.Price) AS year\_avg\_income

FROM

RESERVATIONS AS R

JOIN EXTRASOFRESERVATION AS EXR ON EXR.ReservationID = R.ReservationID

JOIN RESULTS AS RE ON RE.ResultID = EXR.ResultID

JOIN CARS AS C ON C.CarID = RE.CarID

JOIN SEARCHES AS S ON S.SearchIP = RE.SearchIP AND S.SearchDT = RE.SearchDT

JOIN EXTRAS AS E ON E.SearchIP = S.SearchIP AND E.SearchDT = S.SearchDT AND E.ResultID = RE.ResultID

GROUP BY

C.BranchID

HAVING

AVG(RE.pricePerDay \* DATEDIFF(DAY, S.StartDate, S.FinishDate) + EXR.Qty \* E.Price) <

(

SELECT

AVG(RE.pricePerDay \* DATEDIFF(DAY, S.StartDate, S.FinishDate) + EXR.Qty \* E.Price)

FROM

RESERVATIONS AS R

JOIN EXTRASOFRESERVATION AS EXR ON EXR.ReservationID = R.ReservationID

JOIN RESULTS AS RE ON RE.ResultID = EXR.ResultID

JOIN CARS AS C ON C.CarID = RE.CarID

JOIN SEARCHES AS S ON S.SearchIP = RE.SearchIP AND S.SearchDT = RE.SearchDT

JOIN EXTRAS AS E ON E.SearchIP = S.SearchIP AND E.SearchDT = S.SearchDT AND E.ResultID = RE.ResultID

)

ORDER BY

year\_avg\_income DESC;

**סיכום האופטימיזציות שבוצעו ע"י הכלי:**

**1. מניעת חישובים כפולים:**

בשאילתה הראשוניתהחישוב של הממוצע (AVG) נעשה פעמיים: גם בHAVING וגם בתת-השאילתה.

**2. שימוש ב-CTE:**פיצול השאילתה למקטעים לוגיים:

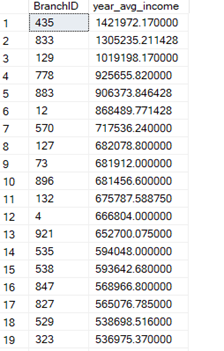
* + - IncomeData מחשב את הממוצע השנתי (year\_avg\_income) עבור כל סניף.
    - OverallAverageIncome מחשב את הממוצע הכללי (overall\_avg\_income) על בסיס הנתונים מ-IncomeData
  + מאפשר קריאות טובה יותר וניהול קל של השאילתה.

**3. אינדקסים מותאמים:**  
המטרה היא לשפר ביצועים של פעולות JOIN, GROUP BY ושליפת נתונים.

קישור לשיחה עם הCHATGPT :

<https://chatgpt.com/share/67952df0-66b8-800b-bb5b-f6b6f82d749c>

פלט השאילתה:



* השאילתה השנייה שבחרנו לייעל הינה השאילתה הבאה:

SELECT S.SearchDT,S.SearchIP,S.StartDate,S.FinishDate,S.BranchT,S.BranchR,R.ResultID, E.Extra

FROM SEARCHES AS S LEFT JOIN RESULTS AS R ON S.SearchIP = R.SearchIP

AND S.SearchDT = R.SearchDT LEFT JOIN

EXTRAS AS E ON R.SearchIP = E.SearchIP

AND R.SearchDT = E.SearchDT

AND R.ResultID = E.ResultID LEFT JOIN

EXTRASOFRESERVATION AS ER ON E.SearchIP = ER.SearchIP

AND E.SearchDT = ER.SearchDT

AND E.ResultID = ER.ResultID

AND E.Extra = ER.Extra

WHERE ER.ReservationID IS NULL  
ORDER BY S.SearchDT DESC

השאילתה לאחר אופטימיזציה של הבינה המלאכותית:

-- יצירת אינדקסים

CREATE INDEX idx\_searches\_search ON SEARCHES(SearchIP, SearchDT, StartDate, FinishDate, BranchT, BranchR);

CREATE INDEX idx\_results\_search ON RESULTS(SearchIP, SearchDT, ResultID);

CREATE INDEX idx\_extras\_search ON EXTRAS(SearchIP, SearchDT, ResultID, Extra);

CREATE INDEX idx\_extrasofreservation\_search ON EXTRASOFRESERVATION(SearchIP, SearchDT, ResultID, Extra, ReservationID);

-- שאילתה אופטימלית

SELECT

S.SearchDT,

S.SearchIP,

S.StartDate,

S.FinishDate,

S.BranchT,

S.BranchR,

R.ResultID,

E.Extra

FROM

SEARCHES AS S

LEFT JOIN

RESULTS AS R

ON

S.SearchIP = R.SearchIP

AND S.SearchDT = R.SearchDT

LEFT JOIN

EXTRAS AS E

ON

R.SearchIP = E.SearchIP

AND R.SearchDT = E.SearchDT

AND R.ResultID = E.ResultID

WHERE

NOT EXISTS (

SELECT 1

FROM EXTRASOFRESERVATION AS ER

WHERE

E.SearchIP = ER.SearchIP

AND E.SearchDT = ER.SearchDT

AND E.ResultID = ER.ResultID

AND E.Extra = ER.Extra

)

ORDER BY

S.SearchDT DESC;

קישור לשיחה עם הבינה המלאכותית:

<https://chatgpt.com/share/679380df-e298-800f-9b59-55b1c87b9efb>

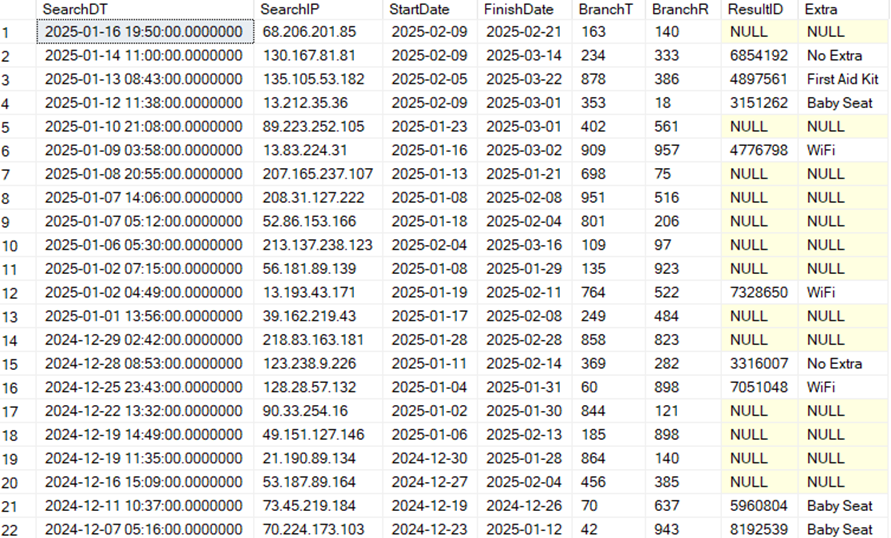
**הצעות לאופטימיזציה:**

**1. שימוש ב-EXISTS במקום JOIN:**לעיתים קרובות, שימוש ב-EXISTS במקום צירופים מיותרים יכול לשפר את הביצועים.   
כך נוכל למנוע איחוד טבלאות שאיננו הכרחי ולהתמקד בסינון הרשומות.  
עם LEFT JOIN: השאילתה מבצעת צירוף של שתי הטבלאות ואז מסננת את התוצאות. כל הנתונים מהצירוף נשמרים בזיכרון, גם אם הם לא נחוצים.  
עם NOT EXISTS: השאילתה רק בודקת אם קיימת התאמה, והיא מפסיקה ברגע שמוצאת רשומה מתאימה, ללא צירוף מלא.

**2. הוספת אינדקסים:**ודא שהעמודות שמשמשות בתנאים של הצירופים והסינון (כמו SearchIP, SearchDT, ResultID ,Extra ) מכוסות על ידי אינדקסים.

**3. השימוש ב- SELECT 1:**   
עשוי להיות טיפה יותר מהיר מכיוון שאינו צריך לקרוא או לעבד נתונים ספציפיים מהטבלה. עם זאת, ברוב מערכות ה-SQL ההבדל בביצועים הוא זניח. בעצם SELECT 1 הוא יותר "מפורש" ומרמז למנוע ה-SQL שאתה לא באמת צריך מידע מתוך הטבלה הפנימית, אלא רק את קיום הרשומה.

פלט השאילתה :



פרק שני - בונוס

מה שהטכניקה עושה זה בעצם המרת שורות (בקוד שלנו נתונים של כל חודש) לעמודות. זה יכול להיות מאוד שימושי אם אנחנו רוצים לראות את הנתונים בצורה שמקלה על הקריאה וההשוואה בין חודשים שונים, במקום שיהיה עלינו לחפש את כל המידע בתוך שורות נפרדות.  
הקוד משתמש ב-VIEW שייצרנו במטלה 3 להצגת הנתונים דרך הPowerBI.  
ומציג לנו את כלל הסניפים ואת ההכנסה שלהם עבור כל חודש ב2024 בתצוגה שנעימה וברורה לעין.  
איך זה עובד?

* + - 1. **נתונים מקוריים:** יש לנו טבלה (כמו ReservationRevenue) בה כל שורה מייצגת הכנסה של סניף בחודש מסוים.
      2. **הפיכת שורות לעמודות:** באמצעות PIVOT, אנו לוקחים את הנתונים מהשדות MonthOfReservation ו-TotalRevenue וממירים אותם לעמודות, כך שכל חודש מקבל עמודה משלו.   
         השיטה הזו מאפשרת הצגה נוחה יותר של נתונים חודשיים.

**הקוד למימוש:**

SELECT

BranchID,

January=COALESCE([1], 0),

February=COALESCE([2], 0),

March=COALESCE([3], 0),

April=COALESCE([4], 0),

May=COALESCE([5], 0),

June=COALESCE([6], 0),

July=COALESCE([7], 0),

August=COALESCE([8], 0),

September=COALESCE([9], 0),

October=COALESCE([10], 0),

November=COALESCE([11], 0),

December=COALESCE([12], 0)

FROM (

SELECT BranchID,MonthOfReservation,TotalRevenue

FROM ReservationRevenue

WHERE YearOfReservation = 2024

) AS SourceData

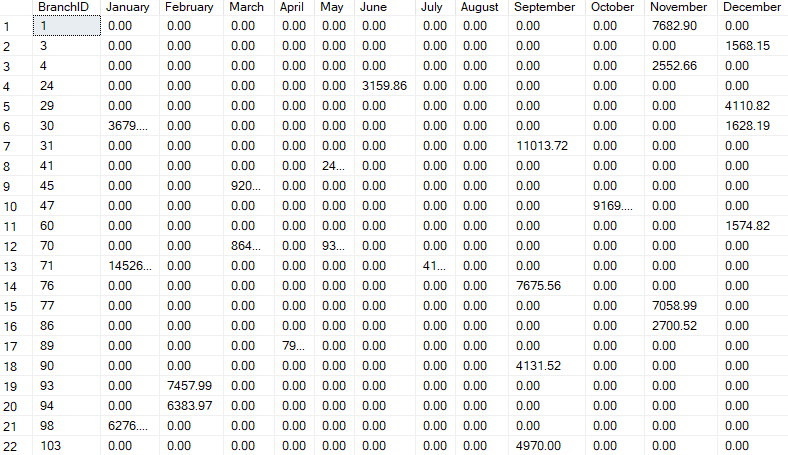
PIVOT (

SUM(TotalRevenue)

FOR MonthOfReservation IN ([1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12])

) AS PivotedData;

**פלט השאילתה:**



טבלאות זמניות [**Temporary Tables**](https://www.sqlservertutorial.net/sql-server-basics/sql-server-temporary-tables/)**)) :**טכניקת טבלאות זמניות ב-SQL היא כלי שימושי במיוחד לניהול תוצאות ביניים או נתונים שדורשים עיבוד חוזר לאורך חיי סשן או פרוצדורה. היא מאפשרת ליצור טבלאות שמבנה הנתונים שלהן, או התוכן שלהן, מוגדר באופן דינמי בזמן ריצה, כאשר הנתונים הללו זמינים רק למשך זמן מוגבל.  
  
**מתי כדאי להשתמש בטבלאות אלו:**

* עיבוד שאילתות מורכבות
* שיפור ביצועים
* צמצום כפילויות
* שימוש חוזר בנתונים

הגיון עסקי: ממוצע של עלות ההשכרה לרכבים מסניף מסוים עבור כל התוצאות חיפושים. נשתמש בטבלה זמנית כדי לאחסן את הנתונים הבסיסיים לעיבוד נוח.  
 נבחר טווח תאריכים מסוים וסניף מסוים ונבדוק מה הממוצע ההשכרות בתאריכים אלו. מידע זה יעזור לנו להבין אם יש חודש עם פחות השכרות אולי בגלל הממוצע הגבוה ונוכל לעשות הנחה מסוימת.

CREATE TABLE #TempSearchResults (

BranchID INT,

SearchIP VARCHAR(20),

SearchDT DATETIME,

CarID VARCHAR(17),

PricePerDay DECIMAL(10, 2)

);

-- הכנסת נתונים לטבלה הזמנית

INSERT INTO #TempSearchResults (BranchID, SearchIP, SearchDT, CarID, PricePerDay)

SELECT

S.BranchT AS BranchID,

S.SearchIP,

S.SearchDT,

R.CarID,

R.PricePerDay

FROM

SEARCHES AS S

JOIN RESULTS R ON S.SearchIP = R.SearchIP AND S.SearchDT = R.SearchDT

WHERE

S.BranchT = 200 -- לדוגמה, סינון לפי סניף מס' 1

AND S.StartDate >= '2024-01-01' AND S.FinishDate <= '2024-12-31';

-- חישוב עלות ממוצעת לרכבים בסניף זה

SELECT

BranchID,

AVG(PricePerDay) AS AvgPricePerDay

FROM

#TempSearchResults

GROUP BY

BranchID;

-- מחיקת הטבלה הזמנית בסיום

DROP TABLE IF EXISTS #TempSearchResults;

פלט השאילתה:

