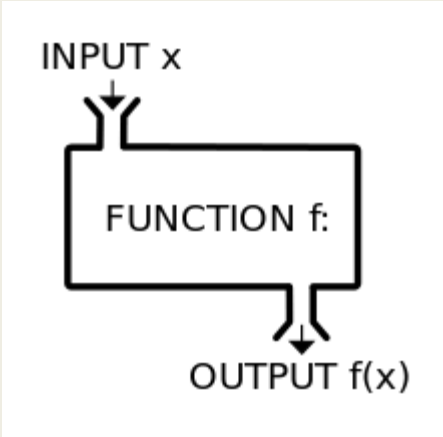
מבוא לבסיסי נתונים ומערכות מידע

מצגת 8 – פונקציות, שאילתות JOIN

דביר מיטב

פונקציות

- בשיעור שעבר למדנו על שדות מחושבים בשאילתות, וכי ניתן לחשב את הערכים של שדות מחושבים בשאילתות באמצעות פונקציות הקבצה.
- קיימות פונקציות נוספות ב-Access אשר ניתן לעשות בהן שימוש לצורך בניית שדות מחושבים.
- בדומה לפעולות שאתם מכירים מלימודי יסודות מדעי המחשב, פונקציות ב-Access יכולות לקבל ערך כפרמטר, ולהחזיר ערך (לדוגמה: פונקציה המקבלת כפרמטר תאריך ומחזירה את השנה של התאריך).
- ב-Access יש רשימה של פונקציות, המסווגות לפי סוג הנתונים עליהן הן פועלות (מספר, תאריך וכו').
- כמו בשפת Java, אם הפונקציה מקבלת פרמטרים (הערכים עליהם היא פועלת), הם ירשמו בתוך סוגריים עגולים. אם הפונקציה לא מקבלת פרמטרים, הסוגריים יהיו ריקים.
- הפונקציות אשר עליהן נלמד היום **<u>אינן</u>** פונקציות הקבצה (כלומר אינן דורשות לעשות שימוש ב-GROUP BY).
- נהוג לכתוב את שמות הפונקציות **באותיות גדולות באנגלית**.



פונקציות

- להלן טבלה עם הפונקציות אשר השימוש בהן הוא הנפוץ ביותר ב-Access:

שם הפונקציה	תיאור הערך המוחזר	טיפוס הפרמטר
LEN (text)	מחזירה את מספר התווים שמכיל הטקסט	טקסט
CSTR(num)	הופכת את המספר num לטקסט	מספר
INT(num)	מקבלת מספר עשרוני ומחזירה את החלק השלם. למשל: INT (4.678) ← 4	מספר עשרוני
ABS(num)	מחזירה את ערכו המוחלט של המספר למשל: ABS|(-54) ← 54	מספר
DATE()	מחזירה את התאריך הנוכחי	ללא פרמטרים
NOW()	מחזירה את התאריך הנוכחי כולל השעה	ללא פרמטרים
DAY(aDate)	מקבלת תאריך ומחזירה את היום בחודש (מספר בין 1 ו- 31)	תאריך
MONTH(aDate)	מקבלת תאריך ומחזירה את החודש (מספר בין 1 ו- 12)	תאריך
YEAR(aDate)	מקבלת תאריך ומחזירה את השנה ב- 4 ספרות	תאריך

דוגמה לשימוש בפונקציות

- בשיעור שעבר ראינו את הדוגמה לשאילתה שממירה את השכר של מורים משקלים לדולרים.

```
SELECT    name, salary, salary/3.9 AS שכר בדולרים
FROM      tblTeachers
```

- אם אנו מעוניינים לקבל תוצאה שהיא מספר שלם, ניתן לעשות שימוש בפעולה INT, שפעולת באותו האופן כמו המרה ל-int בשפת Java (עיגול כלפי מטה):

```
SELECT name, salary, salary/3.9 AS SalaryDollars1, INT(salary/3.9) AS SalaryDollars2
FROM tblTeachers
```

SalaryDollars1	SalaryDollars2
1769.23	1769.00
1410.26	1410.00
2102.56	2102.00
1205.13	1205.00
1153.85	1153.00
1666.67	1666.00
1128.21	1128.00
2435.90	2435.00
1974.36	1974.00
1641.03	1641.00
2051.28	2051.00
1666.67	1666.00

שני האפסים אחרי הנקודה העשרונית עדיין מופיעים מאחר ובפורמט של השדה נבחר להציג שתי ספרות אחרי הנקודה העשרונית

פונקציות המטפלות בתאריכים

- הפונקציות החשובות והשימושיות ביותר מבין הפונקציות בטבלה הן הפונקציות אשר מטפלות בתאריכים.

DATE()	מחזירה את התאריך הנוכחי	ללא פרמטרים
NOW()	מחזירה את התאריך הנוכחי כולל השעה	ללא פרמטרים
DAY(aDate)	מקבלת תאריך ומחזירה את היום בחודש (מספר בין 1 ו- 31)	תאריך
MONTH(aDate)	מקבלת תאריך ומחזירה את החודש (מספר בין 1 ו- 12)	תאריך
YEAR(aDate)	מקבלת תאריך ומחזירה את השנה ב- 4 ספרות	תאריך

- דוגמה לשאילתה שעושה שימוש בתאריך יום ההולדת כדי לחשב את הגיל של המשתמש, מספר היום שבו נולד, מספר החודש שבו נולד והשנה שבה נולד:

```
SELECT name, birthday, YEAR(DATE())-YEAR(birthday) AS Age,
YEAR(birthday) as Year, MONTH(birthday) AS Month,
DAY(birthday) AS Day
FROM tblUsers
```

שימו לב כי זו אינה צורה מדויקת לחשב את הגיל של המשתמש

AGE	YEAR	MONTH	DAY	birthDay	שם
35	1975	2	1	01/02/1975	אבי
25	1985	4	25	25/04/1985	יוסי
10	2000	1	1	01/01/2000	גדי
1	2009	12	31	31/12/2009	דני

הפרש בין תאריכים

- תוצאת חישוב שלצ הפרש בין שני תאריכים הינה מספר הימים בין שני תאריכים אלו.
- לדוגמה, כדי לחשב את הגיל של המשתמש בימים, ניתן לעשות שימוש בשאילתה:

```
SELECT name, birthday, Date() – birthday AS הגיל בימים
FROM tblUsers
```

הגיל בימים
12835
9099
3735
83

- ניסיון לחסר את התאריך המאוחר יותר מהתאריך הנוכחי יחזיר מספר שלילי.
- כדי לחשב תאריך בעוד מספר כלשהו של ימים, ניתן להוסיף מספר שלם של ימים לתאריך. תוצאת החיבור תהיה התאריך החדש אשר מתקבל לאחר הוספת מספר הימים.
- דוגמה לשאילתה שמוסיפה 10 ימים לתאריך יום ההולדת של המשתמש:

```
SELECT name, birthday, birthday+10 AS birthday+10
FROM tblUsers
```

birthDay	birthDay+10
01/02/1975	11/02/1975
25/04/1985	05/05/1985
01/01/2000	11/01/2000
31/12/2009	10/01/2010

IIF – Immediate If

- פונקציה נוספת עליה נלמד במסגרת מצגת זו היא הפונקציה IIF.
- הפונקציה IIF דומה במעט להוראת התנאי המוכרת משפת Java, אך בשונה מהוראת התנאי היא מקבלת שלושה פרמטרים: התנאי, הערך של השדה אם התנאי יתקיים, והערך של השדה עם התנאי לא יתקיים.
- מבנה הפונקציה IIF הינו:

IIF (תנאי , ערך אם התנאי מתקיים , ערך אם התנאי לא מתקיים)

- לדוגמה, כדי לממש את השאילתה שראינו במצגת הקודמת, אשר מציגה את ציוני התלמידים אחרי תוספת של 10% לציון, בתנאי שהציון לא יעבור את ה-100, ניתן לעשות שימוש ב-IIF.
- משפט ה-IIF יהיה:
אם (ציון גבוה מ-90, הצג את הציון 100, הוסף 10% לציון של התלמיד).

IIF ([grade]>90 , 100 , INT ([grade]*1.1 + 0.5))

עיגול לשלם הקרוב ביותר ע"י הוספת 0.5 ועיגול כלפי מטה

לדוגמה:

INT (51 * 1.1 + 0.5) → int (56.1 + 0.5) → int (56.6) → 56
INT (78 * 1.1 + 0.5) → int (85.8 + 0.5) → int (86.3) → 86

IIF – Immediate If

- דוגמה נוספת לשימוש ב-IIF היא בשאילתה שמציגה הודעה לתלמיד שהציון שלו בבחינה כלשהי הוא נכשל כי עליו לגשת למבחן במועד ב'.

```
IIF ([grade] < 55 , "עליך לגשת למועד ב'" , "" )
```

- בדוגמה זו, ההודעה של "עליך לגשת למועד ב'" תוצג רק לתלמיד שהציון שלו במבחן נמוך מ-55. אם הציון של התלמיד גדול או שווה ל-55, תוצג לתלמיד מחרוזת ריקה בשדה הרלוונטי (בדומה למחרוזת הריקה שאתם מכירים מ-Java).
- דוגמאות נוספות:

```
= iif (IsNull( [Region] ), [City]&" "& [PostalCode], [City]&" "&[Region]&" " &[PostalCode] )
```

מציג את הערכים של השדות City ו-PostalCode אם Region הוא **Null**; אחרת, הוא מציג את הערכים בשדות City, Region ו-PostalCode.

```
=iif ( IsNull  ( [RequiredDate] - [ShippedDate] ), "Check for a missing date", [RequiredDate] – [ShippedDate] )
```

מציג את ההודעה "Check for a missing date" אם התוצאה של החיסור ShippedDate מ-RequiredDate הוא **Null**; אחרת, הוא מציג את ההפרש בין הערכים בשדות RequiredDate ו-ShippedDate.

```
= iif ( [Confirmed] = "Yes", "Order Confirmed", "Order Not Confirmed")
```

מציג את ההודעה "Order Confirmed" אם הערך של השדה Confirmed הוא Yes; אחרת, הוא מציג את ההודעה "Order Not Confirmed".

```
= iif ( IsNull ( [Country] ), " ",  [Country] )
```

מציג מחרוזת ריקה אם הערך של השדה Country הוא Null, אחרת הוא מציג את הערך של השדה Country.

Null: ערך אותו ניתן להזין בשדה או להשתמש בו בביטויים או בשאילתות כדי לציין נתונים חסרים או לא ידועים.

שאילתה המבוססת על שאילתה

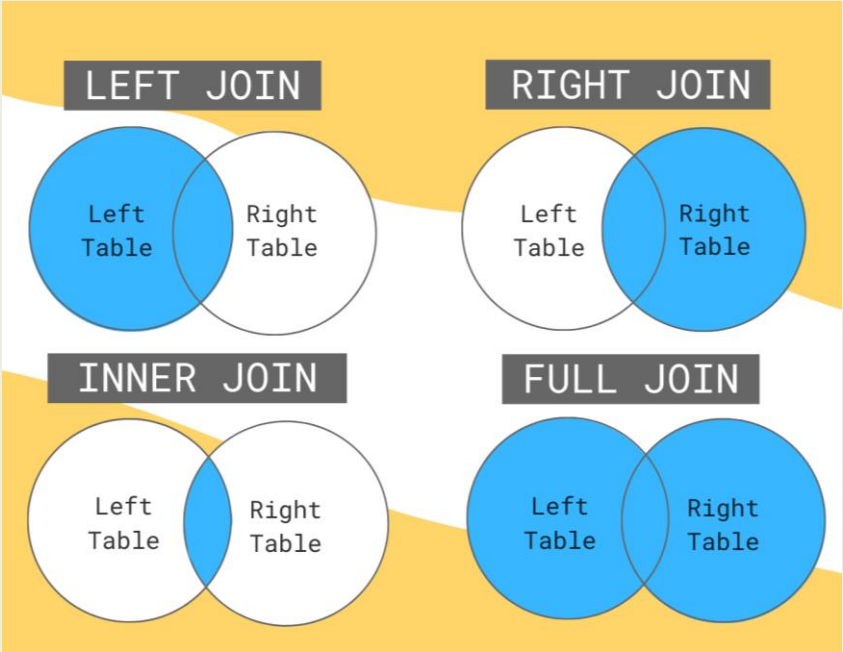
- עד כה ראינו דוגמאות לשאילתות אשר עושות שימוש בנתונים מטבלאות, אך ניתן גם לממש שאילתות אשר עושות שימוש בנתונים משאילתות אחרות שכבר מימשתם.
- בשאילתות המבוססות על שאילתות קיימות, יש לציין בחלק של ה-FROM את שם השאילתה עליה אתם מתבססים:

```
SELECTf ield1, field2...
FROM ExistingQuery
WHERE some_condition;
```

- לדוגמה, כדי לממש שאילתה המציגה את שמות וציוני כל התלמידים בשכבה י' שקיבלו ציון גבוה מהממוצע במקצוע מתמטיקה, יש ליצור תחילה שאילתה אשר תחשב ותציג את הציון הממוצע במתמטיקה לתלמידי כיתות י' (כפי שלמדנו בשיעור שעבר).
- לאחר מכן, יש ליצור שאילתה חדשה שתציג את השמות והציונים של כל התלמידים בשכבה י' שקיבלו ציון גבוה מהממוצע במקצוע מתמטיקה.
- לשם כך, יש צורך לשלב מספר טבלאות באמצעות שאילתות JOIN.
  - *חזרה על כך החל מהשקופית הבאה.*

חזרה על שאילתות עם מספר טבלאות - JOIN

- למדנו כי מסד נתונים מכיל מספר טבלאות (לדוגמה: טבלת ספרים, עובדים, לקוחות).
- לעתים יש צורך בשאילתות שמשלבות נתונים ממספר טבלאות במסד הנתונים.
- כדי לשלב נתונים משתי טבלאות שונות בשאילתה, יש לעשות שימוש במילה השמורה **JOIN**.
- בשורת ה-JOIN, יש לציין מה השדה בטבלה הראשונה שלפיו אנו רוצים לבצע את השילוב בין הטבלאות, ומהו השדה המתאים לו בטבלה השנייה.
  - *השדה בטבלה הראשונה נקרא "מפתח ראשי", ונלמד עליו בשקופית הבאה.*
- קיימים מספר סוגים שונים של שילובים (JOIN) שניתן לבצע בין טבלאות (כפי שניתן לראות בתמונה).



INNER JOIN

- כדי שיהיה ניתן לקשר בין שתי טבלאות, על שתי הטבלאות לכלול שדה של מפתח ראשי שמזהה כל ישות במסד הנתונים באופן ייחודי.
- שימוש ב-JOIN בשאילה מאפשר לשלב שורות משתי טבלאות נפרדות על פי המפתח הראשי.
- הסוג הראשון של JOIN עליו נלמד הוא **INNER JOIN** – תהליך שבו השורות בטבלה הראשונה ובטבלה השנייה שיש להן את אותו המפתח (אשר נקבע ע"י שימוש ב-**ON** בשאילתה) משולבות בשורה אחת שכוללת את כל העמודות מכל אחת משתי הטבלאות.
- ניתן להפעיל על הטבלה שנוצרת ע"י ה-INNER JOIN את כל שאר החלקים של שאילתות ה-SQL שהכרנו לפני כן, כמו WHERE, ORDER BY, LIMIT וכו'.

Select query with INNER JOIN on multiple tables

```
SELECT column, another_table_column,
FROM mytable
INNER JOIN another_table
    ON mytable.id = another_table.id
WHERE condition(s)
ORDER BY column, … ASC/DESC
LIMIT num_limit OFFSET num_offset;
```

דוגמאות לשאילתות עם INNER JOIN

Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000

1. Find the domestic and international sales for each movie

```
SELECT title, domestic_sales, international_sales
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id;
```

2. Show the sales numbers for each movie that did better internationally rather than domestically

```
SELECT title, domestic_sales, international_sales
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id
WHERE international_sales > domestic_sales;
```

3. List all the movies by their ratings in descending order

```
SELECT title, rating
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id
ORDER BY rating DESC;
```

OUTER JOIN

- השימוש ב-INNER JOIN מתאים כאשר אנו מעוניינים שהטבלה שמתקבלת כתוצאה מהרצת השאילתה תכלול רק את הנתונים שקיימים בשתי הטבלאות.
- אם יש לנו שתי טבלאות עם נתונים אסימטריים (כלומר שנתונים על ישות מסוימת קיימים רק באחת מהטבלאות), ניתן לעשות שימוש ב-OUTER JOIN במקום כדי לוודא שלא מאבדים חלק מהמידע בתוצאות של השאילתה.
- קיימים שלושה סוגים של OUTER JOINs: **LEFT JOIN**, **RIGHT JOIN** או **FULL JOIN**.
- בדומה ל-INNER JOIN, גם בכל אחד משלושת הסוגים של ה-OUTER JOIN צריך לציין מהו השדה שלפיו יתבצע הקישור בין הטבלאות (המפתח הראשי).

```
SELECT column, another_column, …
FROM mytable
INNER/LEFT/RIGHT/FULL JOIN another_table
    ON mytable.id = another_table.matching_id
WHERE condition(s)
ORDER BY column, … ASC/DESC
LIMIT num_limit OFFSET num_offset;
```

OUTER JOIN

- כשמקשרים בין טבלה A לטבלה B:
  - ***LEFT JOIN*** *כולל את כל השורות מטבלה A, גם אם אין עבורן שורה מתאימה בטבלה B.*
  - ***RIGHT JOIN*** *כולל את כל השורות מטבלה B, גם אם אין עבורן שורה מתאימה בטבלה A.*
  - ***FULL JOIN*** *כולל את כל השורות משתי הטבלאות, גם אם אין עבורן שורה מתאימה בטבלה האחרת.*
- שימוש בכל אחד משלושת הסוגים של OUTER JOIN עלול לגרום לכך שערכיהם של חלק מהשדות יהיו NULL.

```
SELECT column, another_column, …
FROM mytable
INNER/LEFT/RIGHT/FULL JOIN another_table
    ON mytable.id = another_table.matching_id
WHERE condition(s)
ORDER BY column, … ASC/DESC
LIMIT num_limit OFFSET num_offset;
```

דוגמאות לשאילתות עם OUTER JOIN

Table: Buildings (Read-Only)

Building_name	Capacity
1e	24
1w	32
2e	16
2w	20

Table: Employees (Read-Only)

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8
Artist	Jakob J.	2w	6
Artist	Lillia A.	2w	7
Artist	Brandon J.	2w	7
Manager	Scott K.	1e	9
Manager	Shirlee M.	1e	3
Manager	Daria O.	2w	6

1. Find the list of all buildings that have employees

```
SELECT DISTINCT building
FROM employees;
```

2. Find the list of all buildings and their capacity

```
SELECT *
FROM buildings;
```

3. List all buildings and the distinct employee roles in each building (including empty buildings)

```
SELECT DISTINCT building_name, role
FROM buildings
  LEFT JOIN employees
    ON building_name = building;
```