

# **SQL Database**

## **מODY נתונים רלוונטיים**

TomerBu

# **נושאים להיום:**

יצירת טבלאות  
יחסים גומליות:  
אחד לאחד  
אחד לרבים  
רבים לרבים

**INSERT נתונים - שאלות TO  
פונקציות אגרגציה**

# יצירת DATABASE:

The screenshot shows a PostgreSQL management interface. On the left, there's a tree view of database objects:

- Servers (1)
  - PostgreSQL 17
    - Databases (2)
      - W3schools
      - postgres
    - Login/Group Roles
    - Tablespaces

A context menu is open over the 'Databases' folder, with 'Create' selected. A sub-menu 'Database...' is also visible.

Below the tree view, a modal window titled 'Create - Database' is open. It has tabs for General, Definition, Security, and Parameters, with 'General' selected. In the 'Database' field, the name 'blog\_db' is entered.

A note on the left says 'לחץ ימני בעכבר' (Right-click here).

# הרצת שאלות מול מסד- הנתונים

The screenshot shows a tree view of a PostgreSQL 17 database structure. The 'blog\_db' database is selected and highlighted with a blue selection bar. A context menu is open over the 'blog\_db' entry, listing various database management options.

- Create >
- Delete
- Delete (Force)
- Refresh...
- Restore...
- Backup...
- CREATE Script
- Disconnect from database
- ERD For Database
- Maintenance...
- Grant Wizard...
- Search Objects...
- PSQL Tool
- Query Tool** (highlighted in dark blue)
- Properties

Left sidebar navigation:

- PostgreSQL 17
  - Databases (3)
    - W3schools
    - blog\_db** (selected)
  - Tablespaces
  - Login/Group Roles
  - Extensions
  - Foreign Data V
  - Languages
  - Publications
  - Schemas
  - Subscriptions
  - postgres

# שאילתת ליצירת טבלה:

```
DROP TABLE IF EXISTS Students;
```

```
CREATE TABLE Students(  
    StudentID INTEGER,  
    FirstName VARCHAR(20),  
    LastName VARCHAR(20)  
);
```

מספר שלם

טקסט באורך משתנה  
מוגבל ל 20 תווים

# הכנסת שורות

```
DROP TABLE IF EXISTS Students;
```

```
CREATE TABLE Students(  
    StudentID INTEGER,  
    FirstName VARCHAR(20),  
    LastName VARCHAR(20)  
) ;
```

-- fill the table with data:

```
INSERT INTO Students(StudentId, FirstName, LastName)  
VALUES (1, 'Moshe', 'Doe'), (2, 'Dave', 'Green');
```

```
SELECT * FROM Students;
```

**בעה עם הטבלה שלנו:  
אין עמודת מפתח ואפשר להכניס רק חלק מהעמודות:**

```
INSERT INTO Students(FirstName)
VALUES ('Shultz');

SELECT * FROM Students;
```

studentid integer	firstname character varying (20)	lastname character varying (20)
1	Moshe	Doe
2	Dave	Green
[null]	Shultz	[null]

# NOT NULL

סינטקס למניעת אי-הכנסה עמודה:

```
CREATE TABLE Students(  
    StudentID INTEGER NOT NULL,  
    FirstName VARCHAR(20) NOT NULL,  
    LastName VARCHAR(20) NOT NULL  
) ;
```

# ID Primary Key

לכל שורה צריכה להיות עמודת מזהה ייחודי  
כך נוכל ליצור יחס גומלין בין הטעלאות.

ברגע שנגיד רעמודה מסויימת בעמודת מפתח:  
**הhiposh lepi haumoda hazat yihya yotter yeil**

המערכת תיצור לנו אינדקס

```
DROP TABLE IF EXISTS Students;

CREATE TABLE Students(
    StudentID INTEGER PRIMARY KEY NOT NULL,
    FirstName VARCHAR(20) NOT NULL,
    LastName VARCHAR(20) NOT NULL
);

-- fill the table with data:
INSERT INTO Students(StudentId, FirstName, LastName)
VALUES (1, 'Moshe', 'Doe'), (2, 'Dave', 'Green');

SELECT * FROM Students;
```

# SERIAL

שדה שמסומן  
Serial

```
DROP TABLE IF EXISTS Students;
```

```
CREATE TABLE Students(  
    StudentID SERIAL PRIMARY KEY NOT NULL,  
    FirstName VARCHAR(20) NOT NULL,  
    LastName VARCHAR(20) NOT NULL  
) ;
```

-- fill the table with data:

```
INSERT INTO Students(FirstName, LastName)  
VALUES ('Moshe', 'Doe'), ('Dave', 'Green');
```

```
INSERT INTO Students(FirstName, LastName)  
VALUES ('Joe', 'Krup');
```

```
SELECT * FROM Students;
```

המערכת תטפל בערך שלו:  
שיהיה מספר שלם ורץ

המערכת תקבע את ה ID הבא

שאילתת INSERT  
ללא הכנסת ID  
(המערכת מטפלת בזה)

# מיני תרגיל:

1

צרו טבלת מוצרים:

לכל מוצר יש:

ProductID, ProductName, Price

(Serial, PK, NOT NULL)

מספר שלם

2

הזינו 2 מוצרים לבחירתכם.

# פתרונות לתרגיל וTYPE

```
CREATE TABLE Products(  
    ProductID SERIAL PRIMARY KEY NOT NULL,  
    ProductName VARCHAR(64) NOT NULL,  
    Price DECIMAL(10, 2) NOT NULL  
) ;
```

מספר עשרוני

```
INSERT INTO Products(ProductName, Price)  
VALUES('Notebook', 5.9), ('Banana', 3.9);
```

```
SELECT * FROM Products;
```

# דוקומנטציה:

The screenshot shows a Chrome browser window with the title "PostgreSQL: Documentation". The URL in the address bar is "postgresql.org/docs/current/datatype-numeric.html". The page content is about numeric types in PostgreSQL, specifically section 8.1. The page includes navigation links "Prev" and "Up", and a table titled "Table 8.2. Numeric Types" comparing various numeric data types based on storage size, description, and range.

**8.1. Numeric Types**

- 8.1.1. Integer Types
- 8.1.2. Arbitrary Precision Numbers
- 8.1.3. Floating-Point Types
- 8.1.4. Serial Types

מספר עשרוני

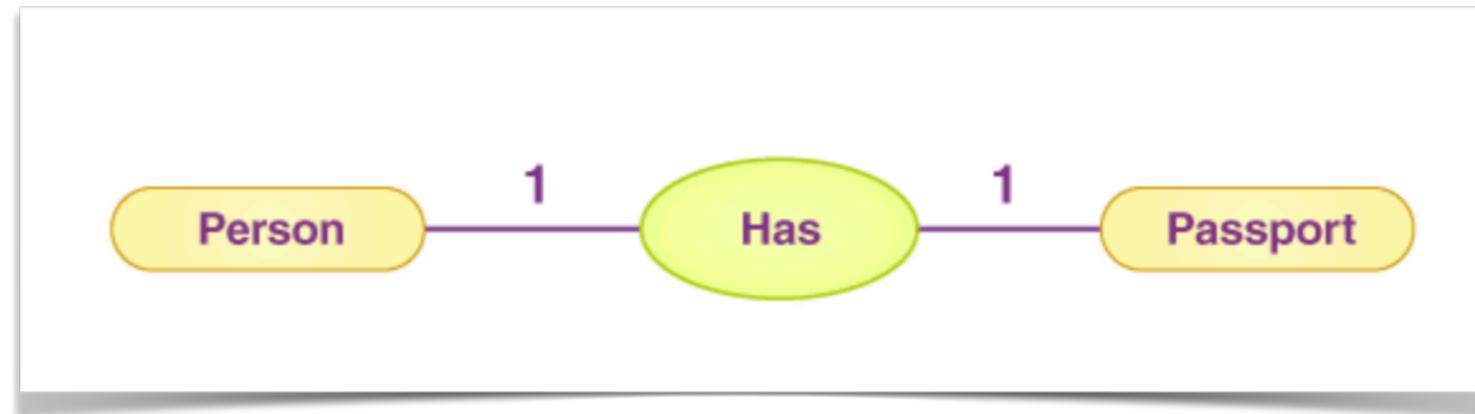
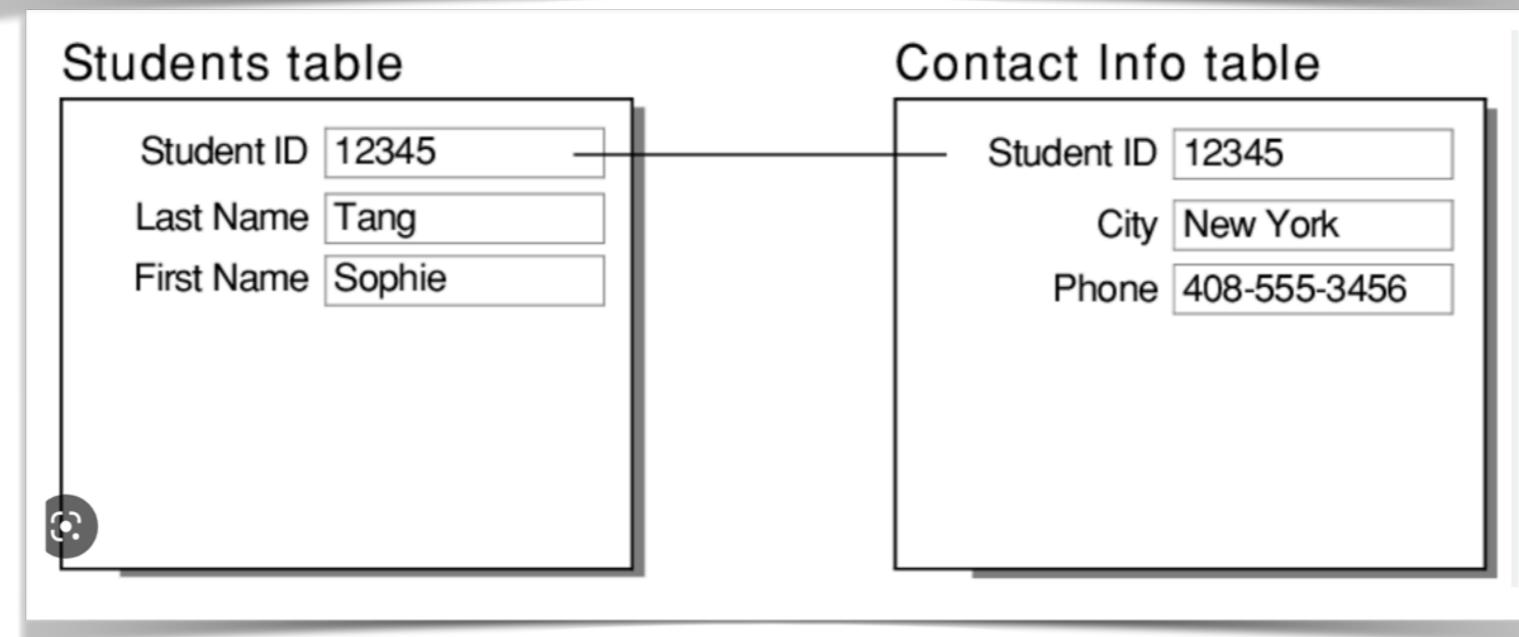
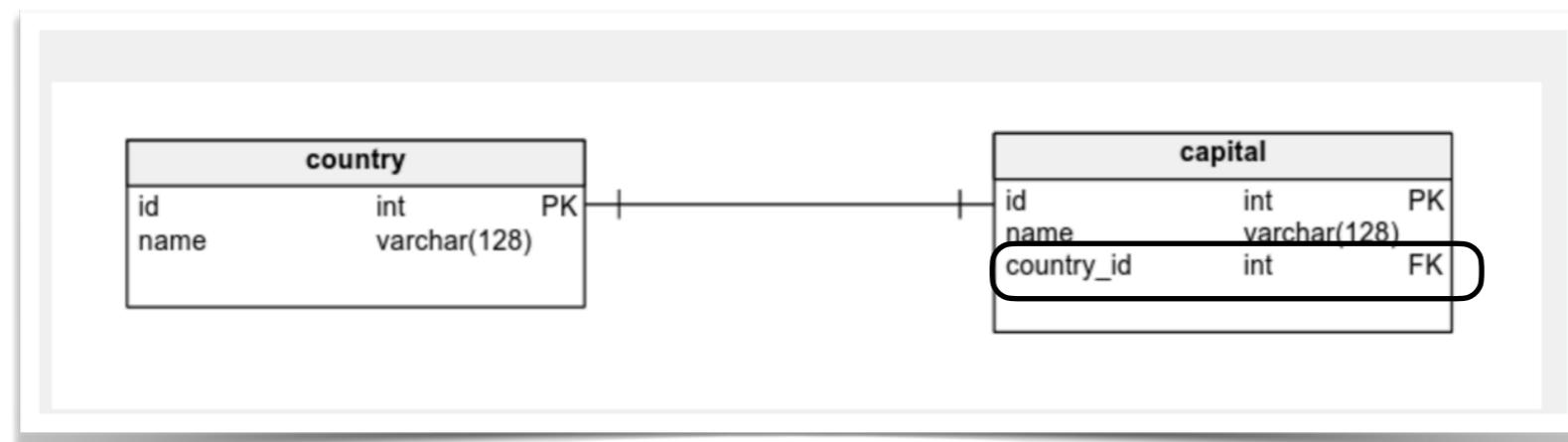
Numeric types consist of two-, four-, and eight-byte integers, four- and eight-byte floating-point numbers, and arbitrary precision numbers.

**Table 8.2. Numeric Types**

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

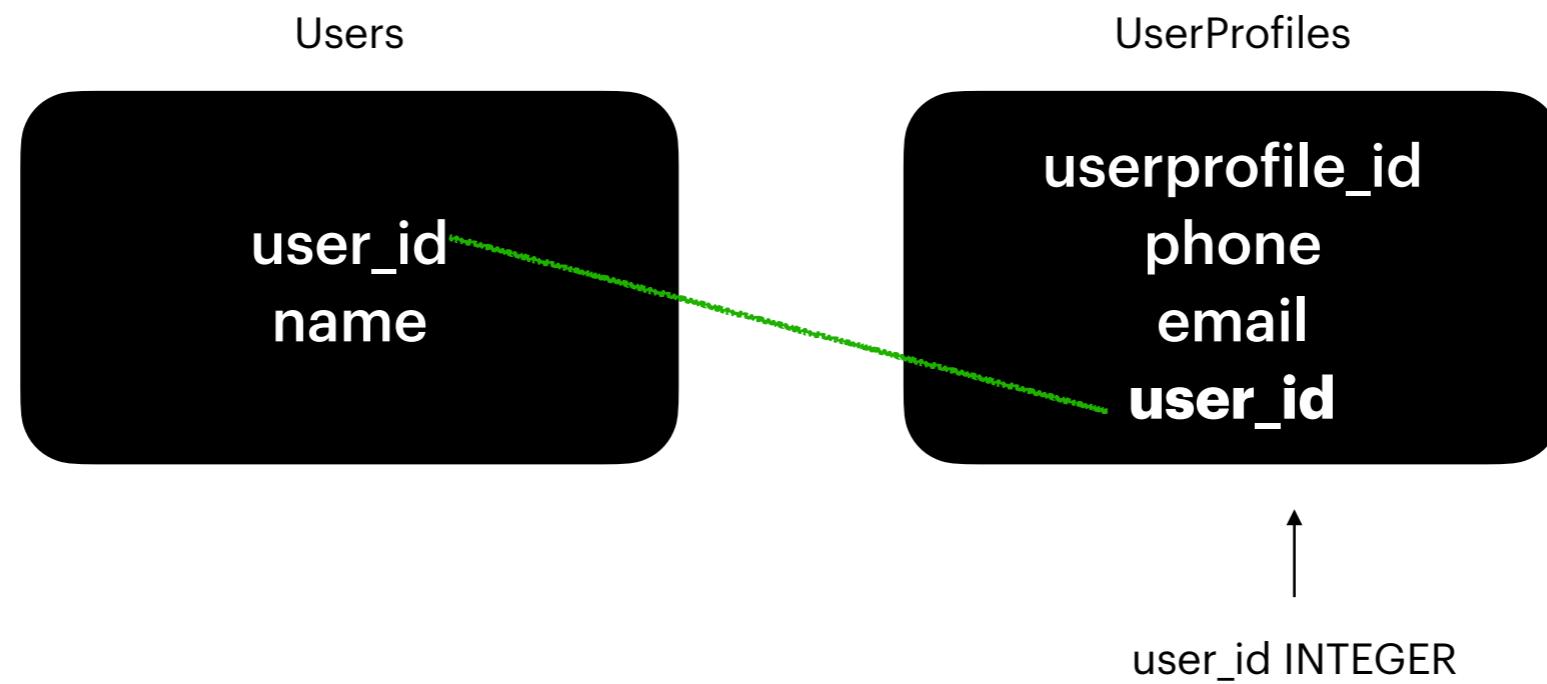
# קשרי גומלין

## מפתח זר לטבלה אחרת



# קשרי גומלין

## מפתח זר לטבלה אחרת



**מינימליסטי:**  
**צרו את הטעלאות המתוארכות בשרטוט**

# קשרי גומלין

## מפתח זר לטבלה אחרת

```
DROP TABLE IF EXISTS Users;  
DROP TABLE IF EXISTS UserProfiles;
```

```
CREATE TABLE Users(  
    UserID SERIAL PRIMARY KEY,  
    Username VARCHAR(32)  
) ;
```

```
CREATE TABLE UserProfiles(  
    UserProfileID SERIAL PRIMARY KEY,  
    Phone VARCHAR(20) NULL,  
    Email VARCHAR(100) NOT NULL,  
    UserID INTEGER  
) ;
```

```
INSERT INTO Users(Username)  
VALUES ('Larry');
```

```
INSERT INTO UserProfiles>Email, UserId  
VALUES('LarryBird@gmail.com', 1);
```

```
SELECT *  
FROM Users JOIN UserProfiles USING(UserID);
```

זה עובד  
אבל יש עוד הרבה אפשרויות  
וליהן נלמד אחרי ההפסקה

# קשרי גומליון

## מפתח זר לטליה אחרת

הכנסת משתמש:

id = 1

הכנסת משתמש חדש

id = 2

הכנסת פרטי משתמש:  
email = 'abc@gmail.com'  
userid = 555

בעיה - אפשר להכניס כל מספר לuserid  
כ噫 הגדרנו אותו כ"כל מספר" INTEGER

# קשרי גומליון

## אלוץ KEY

### ניתן להכניס כל מספר שלם "קיימים בטבלת היחס"

```
DROP TABLE IF EXISTS Users;  
DROP TABLE IF EXISTS UserProfiles;
```

```
CREATE TABLE Users(  
    UserID SERIAL PRIMARY KEY,  
    Username VARCHAR(32)  
) ;
```

```
CREATE TABLE UserProfiles(  
    UserProfileID SERIAL PRIMARY KEY,  
    Phone VARCHAR(20) NULL,  
    Email VARCHAR(100) NOT NULL,  
    UserID INTEGER,  
    FOREIGN KEY (UserID) REFERENCES Users(UserID)  
) ;
```

```
INSERT INTO Users(Username)  
VALUES ('Larry'), ('Jerry'), ('Margo');
```

```
INSERT INTO UserProfiles(Email, UserId)  
VALUES('LarryBird@gmail.com', 555);
```

```
SELECT *  
FROM Users JOIN UserProfiles USING(UserID);
```

הטבלה  
והעמודה

יבדוק שקיים משתמש עם id = 555

# קשרי גומלין

```
DROP TABLE IF EXISTS Users CASCADE; -- also delete any related rows in other tables  
DROP TABLE IF EXISTS UserProfiles CASCADE;
```

```
CREATE TABLE Users(  
    UserID SERIAL PRIMARY KEY,  
    Username VARCHAR(32)  
) ;
```

```
CREATE TABLE UserProfiles(  
    UserProfileID SERIAL PRIMARY KEY,  
    Phone VARCHAR(20) NULL,  
    Email VARCHAR(100) NOT NULL,  
    UserID INTEGER,  
    FOREIGN KEY (UserID) REFERENCES Users(UserID)  
) ;
```

```
INSERT INTO Users(Username)  
VALUES ('Larry'), ('Jerry'), ('Margo');
```

```
INSERT INTO UserProfiles>Email, UserId  
VALUES('LarryBird@gmail.com', 1);
```

```
SELECT *  
FROM Users JOIN UserProfiles USING(UserID);
```

במחיקת טבלה

ימחק את הרשומות שתלוויות בה בטבלאות אחרות

# קשרי גומלין

## מפתח זר לטבלה אחרת

Country

country\_id  
name

Capital

capital\_id  
name  
country\_id

צרו את הטבלאות בشرطו:

יש להתייחס לFOREIGN KEY REFERENCES

כך שלא יהיה להכנס עיר בירה למדינה שאינה קיימת  
שם המדינה ושם העיר הם שדות חובה.

לא לשכוח PK ושדות SERIAL

מי שסימן:  
יש להציג  
את המדינות  
כולל ערי הבירה  
לפי סדר אלפבית  
(JOIN)  
(ORDER BY)

מי שסימן:  
יש להכניס איז  
France  
עיר בירה  
ארץ  
UK  
עיר בירה  
London

# קשר גומליו

## One to One

### אחד לאחד

```
DROP TABLE IF EXISTS Country CASCADE;
DROP TABLE IF EXISTS Capital CASCADE;
```

```
CREATE TABLE Country(
    CountryID SERIAL PRIMARY KEY,
    CountyName VARCHAR(64) NOT NULL
);
```

```
CREATE TABLE Capital(
    CapitalID SERIAL PRIMARY KEY,
    CapitalName VARCHAR(64) NOT NULL,
    CountryID INTEGER UNIQUE,
    FOREIGN KEY (CountryID) REFERENCES Country(CountryID)
);
```

```
SELECT * FROM Capital;
```

```
INSERT INTO Country(CountyName)
VALUES ('UK'), ('France'), ('Ukraine');
```

```
INSERT INTO Capital(CountryID, CapitalName)
VALUES (1, 'London'), (2, 'Paris'), (3, 'Kyiv');
```

```
SELECT *
FROM Country JOIN Capital USING(CountryID);
```

בלי המילה UNIQUE  
ניתן יהיה להגדיר מספר ערי בירה לאותה מדינה

עם המילה UNIQUE  
יש קשר של אחד לאחד:

לכל מדינה יש בירה אחת.

# **קשר גומליין**

## **One to One**

## **One To Many**

## **Many To Many**

### **\*אחד לאחד:**

לכל מדינה יש עיר בירה אחת בלבד

לכל משתמש:

יש פרופיל משתמש אחד

### **אחד לרבים:**

لسרט אחד יש דירוג התאמה לצפייה אחד.  
דירוג התאמה לצפייה שיק להרבה סרטים

لسרט אחד יש הרבה דירוגים.  
דירוג שיק לסרט מסוימים.

**שיר שיק למלהין אחד**  
**ללהין יש הרבה שירים**

### **רבים לרבים:**

שחקנו משחק בהרבה סרטים  
בסרט יש הרבה שחקנים

סרט יכול להשתיק להרבה זאנרים  
בזאנר יש הרבה סרטים

# קשר גומלין

## Many To Many

Movie

Category

MovieCategory

**movie\_id**  
**title**

**category\_id**  
**category\_name**

**movieCategoryID**  
**movie\_id**  
**category\_id**

(1, batman)  
(2, snowwhite)

(1, action)  
(2, fantasy)  
(3, comedy)  
(4, scifi)  
(5, cartoon)  
(6, horror)  
(7, horror)  
(8, crime)

(1, 2)  
(1, 8)  
(1, 1)

**מיני תרגיל:**

צרו את הטבלאות בشرطוט

הכניסו 8 קטגוריות  
סרט אחד שמשתיך למספר זנרים

# קשר גומליין

## Many To Many

```
DROP TABLE IF EXISTS Movies CASCADE;
DROP TABLE IF EXISTS Categories CASCADE;
DROP TABLE IF EXISTS MovieCategories CASCADE;

CREATE TABLE Movies(
    MovieID SERIAL PRIMARY KEY,
    Title VARCHAR(100) NOT NULL
);

CREATE TABLE Categories(
    CategoryID SERIAL PRIMARY KEY,
    CategoryName VARCHAR(100) NOT NULL
);

-- many to many
--1) relationship table:
CREATE TABLE MovieCategories(
    MovieCategories SERIAL PRIMARY KEY,
    MovieID INTEGER,
    CategoryID INTEGER,
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID),
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);
```

# קשר גומליין

## Many To Many

```
INSERT INTO Movies(title)
VALUES ('Batman'), ('Spiderman'), ('Shrek');

INSERT INTO CATEGORIES(CategoryName)
VALUES ('action'), ('fantasy'), ('comedy') ,('scifi'), ('cartoon'), ('horror'), ('crime'), ('animation');

--2) insert each movie id and related category ids
INSERT INTO MovieCategories(MovieID, CategoryID)
VALUES (1, 1), (1, 2), (1, 4), (1, 7), (2, 1), (2, 2), (2, 4), (2, 7);

--3) query using joins
SELECT title, categoryname
FROM Movies
JOIN MovieCategories USING(MovieID)
JOIN Categories USING(CategoryID);
```

**קשר גומליין**  
**One to One**  
**One To Many**  
**Many To Many**

**\*אחד לאחד:**

נכלול עמודת מפתח זר ו**UNIQUE**

**\*אחד לרבים:**

נכלול עמודת מפתח זר ב- UNIQUE**

**\*רבים לרבים:**

ኒצ'ור טבלה מקשרת  
ובה 2 מפתחות זרים  
ל2 טבלאות קיימות

בהכנסה נזין לטבלה המקשרת  
מזהה של סרט וקטגוריה

# הגדרת מפתח מורכב: Composite Primary Key

```
CREATE TABLE MovieCategories(  
    MovieID INTEGER,  
    CategoryID INTEGER,  
    PRIMARY KEY (MovieID, CategoryID),  
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID),  
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)  
) ;
```

הדרישה מעמודת מפתח - מזהה ייחודי.

ניתן לבחור מהדטה שלנו שילוב של עמודות מפתח  
**בתנאי שהצירוף ייחודי.**

**יתרונו: אין עמודה מיותרת.**  
**אין בזבוז של שטח בדטה-בייס**

# DELETE FROM מחיקת רשומות

```
DELETE FROM Movies  
WHERE MovieId = 1;
```

```
DELETE FROM Movies
```

פקודת הרסנית - תמחק הכל כי אין

```
DELETE FROM MovieCategories  
WHERE MovieID = 1;
```

# ON DELETE

אם הסרט נמחק מטבלת הסרטים  
יש להסיר את הסרט מהטבלה הנוכחית

```
CREATE TABLE MovieCategories(  
    MovieID INTEGER,  
    CategoryID INTEGER,  
    PRIMARY KEY (MovieID, CategoryID),  
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID) ON DELETE CASCADE,  
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID) ON DELETE CASCADE  
) ;
```

אם הקטגוריה נמחקה  
יש להסיר את הקישור לקטגוריה וסרט מהטבלה הנוכחית

# ON DELETE SET NULL

אם מחקו את המדינה:  
השאר את עיר הבירה  
CountryID = NULL  
עם

```
CREATE TABLE Capital(  
    CapitalID SERIAL PRIMARY KEY,  
    CapitalName VARCHAR(64) NOT NULL,  
    CountryID INTEGER UNIQUE,  
    FOREIGN KEY (CountryID) REFERENCES Country(CountryID) ON DELETE SET NULL  
) ;
```

טוב במקרה שמוחקים משתמש  
ורוצים להשאיר את ההיסטוריה שלו:

לדוגמא תגבות.

## תאריכים:

```
-- Timestamp = Date + Time
CREATE TABLE Meetings(
    MeetingID SERIAL PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    MeetingTime TIMESTAMP NOT NULL
);
```

```
-- Date = Date only (no time)
CREATE TABLE Events(
    EventID SERIAL PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    EventDate DATE NOT NULL
);
```

## תאריכים:

TIMESTAMP = תאריך ושעה

```
DROP TABLE IF EXISTS Meetings;
DROP TABLE IF EXISTS Events;
-- Timestamp = Date + Time
CREATE TABLE Meetings(
    MeetingID SERIAL PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    MeetingTime TIMESTAMP NOT NULL
);
```

הזנה של ערך עם שעה:  
**INSERT INTO Meetings(title, MeetingTime)**  
**VALUES ('Project Kickoff', '2025-02-02 09:30:00'),**  
**('Team Meeting', '2025-02-02') ; -- time defaults to 00:00:00.00**

הזנה של ערך ללא שעה (ברירת מחדל - 12 בלילה):  
**SELECT \***  
**FROM Meetings;**

הזנה של השעה הנוכחית:

```
INSERT INTO Meetings(title, MeetingTime)
VALUES ('Break', NOW());
```

## תאריכים:

תאריך בלבד = DATE

```
DROP TABLE IF EXISTS Events;  
-- Date = Date only (no time)  
CREATE TABLE Events(  
    EventID SERIAL PRIMARY KEY,  
    Title VARCHAR(200) NOT NULL,  
    EventDate DATE NOT NULL  
) ;
```

הכנסה של תאריך ידנית:

```
INSERT INTO Events (Title, EventDate)  
VALUES ('Birthday Party', '2025-03-15');
```

YEAR-MONTH-DAY

-- use the current date

הכנסה של התאריך הנוכחי:

```
INSERT INTO Events (Title, EventDate)  
VALUES ('Yoga Class', NOW());
```

מאוד נוח למילון בSQL

```
SELECT *  
FROM Events;
```

לפי שנה  
ואז חודש  
ואז יום

## הטיפוס הבוליאני:

```
DROP TABLE IF EXISTS Tasks;

CREATE TABLE TASKS(
    TaskID SERIAL PRIMARY KEY,
    TaskName VARCHAR(100) NOT NULL,
    IS_COMPLETE BOOLEAN NOT NULL
);

INSERT INTO Tasks(TaskName, IS_COMPLETE)
VALUES ('Call Mom', FALSE);

SELECT *
FROM Tasks;
```

## ערכי ברירת מחדל:

```
DROP TABLE IF EXISTS Tasks;

CREATE TABLE Tasks(
    TaskID SERIAL PRIMARY KEY,
    TaskName VARCHAR(100) NOT NULL,
    ISComplete BOOLEAN NOT NULL DEFAULT FALSE,
    CreatedAt TIMESTAMP DEFAULT NOW()
);

INSERT INTO Tasks(TaskName)
VALUES ('Call Mom');

SELECT *
FROM Tasks;
```

## עדכון של רשומות:

### UPDATE Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

**UPDATE** Customers

```
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

## עדכון של רשומות:

```
DROP TABLE IF EXISTS Tasks;

CREATE TABLE Tasks(
    TaskID SERIAL PRIMARY KEY,
    TaskName VARCHAR(100) NOT NULL,
    IsComplete BOOLEAN NOT NULL DEFAULT FALSE,
    CreatedAt TIMESTAMP DEFAULT NOW()
);

INSERT INTO Tasks(TaskName)
VALUES ('Call Mom'), ('Drink Coffee'), ('Learn Django'), ('Learn SQL');

UPDATE Tasks
SET IsComplete = TRUE
WHERE TaskID = 1;

SELECT *
FROM Tasks;
```

זכור היבט את WHERE  
אחרת יעדכן את כל הרשומות

# פונקציות צבירה: MAX, MIN, COUNT, SUM, AVG

```
SELECT MAX(Price)  
FROM Products; --263.5
```

```
SELECT MIN(Price)  
FROM Products; --2.5
```

```
SELECT COUNT(Price)  
FROM Products;--77 (count of products)
```

```
SELECT SUM(Price)  
FROM Products; --2222.71
```

```
SELECT ROUND(AVG(Price), 1) --round the avarage to 1 decimal point  
FROM Products;
```

# Alias

## לעמודה (החלפת שם)

```
SELECT MAX(Price) AS MaxPrice, MIN(Price), COUNT(Price), SUM(Price), ROUND(AVG(Price), 1) AS "Avarage Price"  
FROM Products;
```



```
SELECT ROUND(AVG(Price), 1) AS "Avarage Product Price"  
FROM Products;
```



## תרגילים:

צרו מערכת לניהול בלוג:

**Posts:** post\_id, created, title, content, userid

**Comments:** comment\_id, created, content, post\_id, user\_id

**Users:** user\_id, username, email, password(hashed)

**GroupTypes:** ***Admin, Editor, User, VipUser***

group\_id, group\_name

בקבוצה יש הרבה משתמשים (הרבה חברים בקבוצה user)  
משתמש יכול להשתתף להרבה קבוצות

**GroupUsers:** user\_id, group\_id

טבלה מקשרת

Group  
Users  
GroupUsers  
Posts  
Comments

סדר ליצירת הטבלאות: