

מיני פרויקט
בקסיסי נתונים

ניהול מאגר נתונים
באונקברסיטה

תיעוד פרויקט מורחב

ליאור מסס – 315402453

קישור לגיט:

<https://github.com/Lior12321/DBProject2453>

תוכן עניינים:

| | |
|-------|---------------------------------|
| 3 | מבוא |
| 4 | ERD diagram |
| 5 | DSD diagram |
| 6-10 | תקשימי ה SQL |
| 6-7 | • Create Tables - SQL |
| 8 | • Drop tables - SQL |
| 9 | • Insert tables - SQL |
| 10 | • Select all - SQL |
| 11 | הסבר עיצובי של מסד הנתונים |
| 12-14 | שיטות הכנסת הנתונים שנבחרו |
| 12 | • הכנסת נתונים באמצעות פייתון |
| 13 | • הכנסת נתונים באמצעות mockaroo |
| 14 | • הכנסת נתונים מקובץ CSV |
| 15-16 | גיבוי הנתונים |
| 17-18 | שחזור הנתונים |

מבוא:

המערכת תעסוק בניהול מערך נתונים של מוסד אקדמי, ותכלול את הישויות המרכזיות הדרושות לניהול כולל של חיי האקדמיה: סטודנטים, מרצים, קורסים, מחלקות, כיתות לימוד והרשמות לקורסים. כל ישות כוללת תכונות רבות, המאפשרות תיעוד וניהול מפורט של כל התהליכים במערכת.

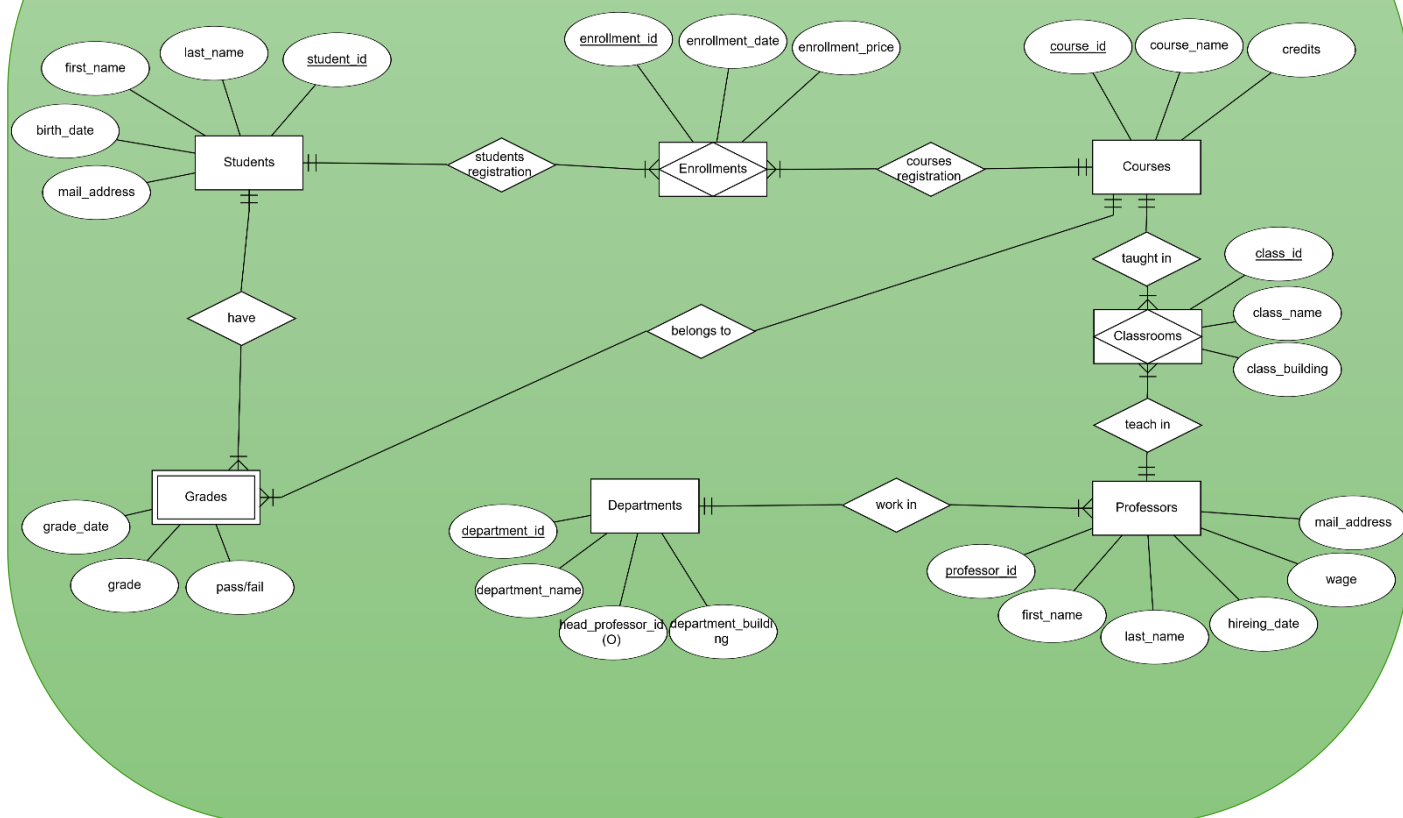
לדוגמה, הישות קורס כוללת מידע על שם הקורס, קוד הקורס, מספר נקודות הזכות והמרצה האחראי; הישות סטודנט כוללת פרטים כמו שם פרטי ומשפחה, מזהה ייחודי, שנת לימודים ומחלקה. באמצעות מבנה זה ניתן לנהל את השיבוץ לקורסים, לקשר בין סטודנטים לקורסים, להזין ציונים, ולתת נתונים אקדמיים.

בהמשך העבודה נציג שאילתת SQL מגוונת אשר יאפשרו לנו:

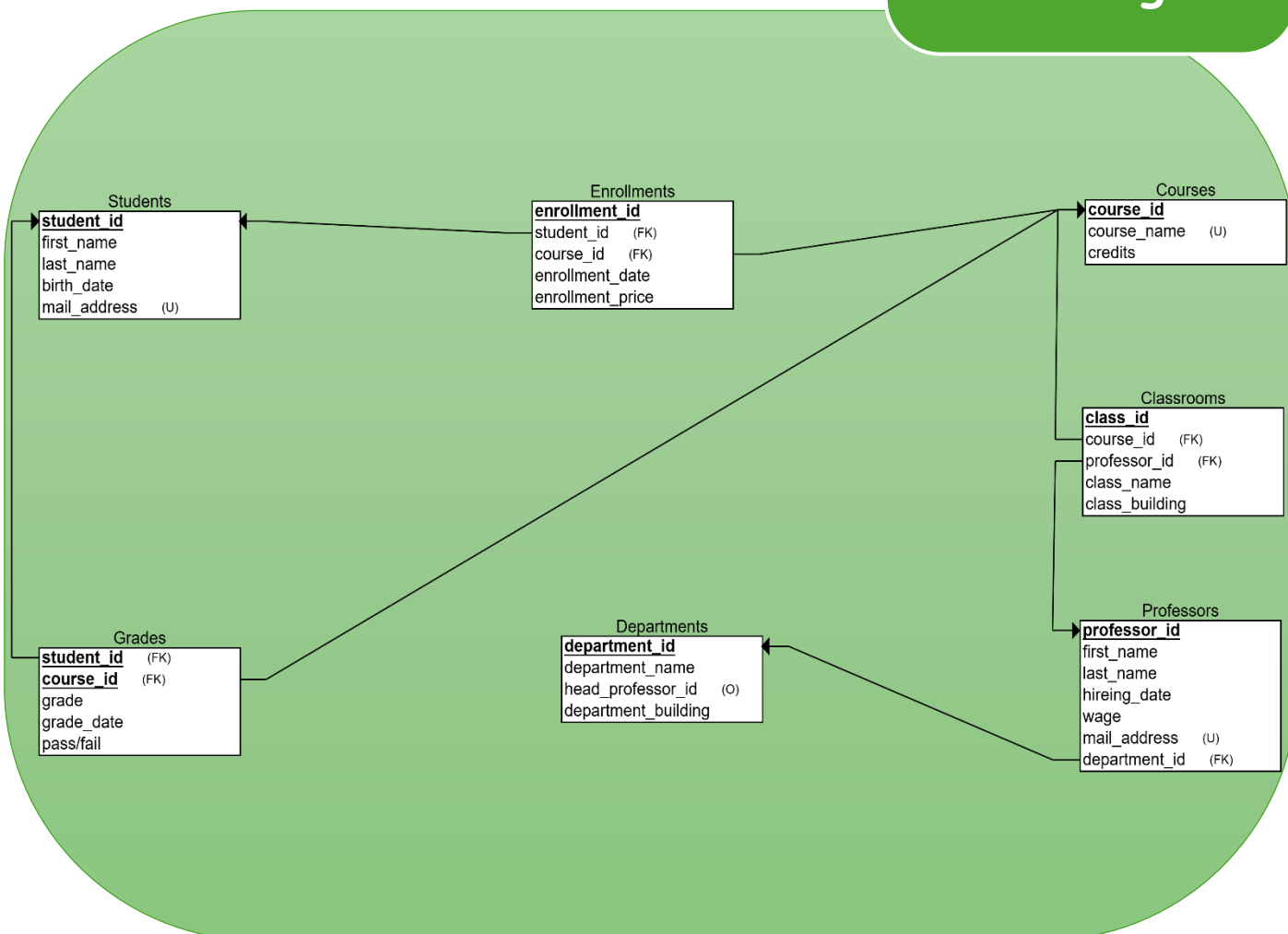
- לשלוף מידע רלוונטי כגון גיליונות ציונים או רשימות סטודנטים לקורסים מסוימים.
- לנהל שינויים בטבלאות – הוספה, עדכון ומחיקה של ישויות.
- לבצע חיתוכים וניתוחים לפי מחלקות, ציונים, קורסים ועוד.

מערכת זו נועדה לשפר את תהליך הניהול של מאגרי המידע האקדמיים, ולהוות בסיס לפיתוח מערכות מידע מתקדמות יותר בעתיד.

:ERD diagram



:DSD diagram



:Create tables - SQL

```

1  CREATE TABLE Students
2  (
3      student_id INT NOT NULL,
4      first_name VARCHAR(50) NOT NULL,
5      last_name VARCHAR(50) NOT NULL,
6      birth_date DATE NOT NULL,
7      mail_address VARCHAR(50) NOT NULL,
8      PRIMARY KEY (student_id),
9      UNIQUE (mail_address)
10 );
11
12 CREATE TABLE Courses
13 (
14     course_id INT NOT NULL,
15     course_name VARCHAR(50) NOT NULL,
16     credits INT NOT NULL,
17     PRIMARY KEY (course_id),
18     UNIQUE (course_name)
19 );
20
21 CREATE TABLE Enrollments
22 (
23     enrollment_id INT NOT NULL,
24     enrollment_date DATE NOT NULL,
25     enrollment_price INT NOT NULL,
26     student_id INT NOT NULL,
27     course_id INT NOT NULL,
28     PRIMARY KEY (enrollment_id),
29     FOREIGN KEY (student_id) REFERENCES Students(student_id),
30     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
31 );
32
33 CREATE TABLE Grades
34 (
35     student_id INT NOT NULL,
36     course_id INT NOT NULL,
37     grade INT NOT NULL,
38     grade_date DATE NOT NULL,
39     pass_fail VARCHAR(4) NOT NULL,
40     PRIMARY KEY (student_id, course_id),
41     FOREIGN KEY (student_id) REFERENCES Students(student_id),
42     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
43 );

```

:Create tables - SQL

```

45 CREATE TABLE Departments
46 (
47     department_id INT NOT NULL,
48     department_name VARCHAR(50) NOT NULL,
49     head_professor_id INT,
50     department_building VARCHAR(50) NOT NULL,
51     PRIMARY KEY (department_id)
52 );
53
54 CREATE TABLE Professors
55 (
56     professor_id INT NOT NULL,
57     first_name VARCHAR(50) NOT NULL,
58     last_name VARCHAR(50) NOT NULL,
59     hiring_date DATE NOT NULL,
60     wage INT NOT NULL,
61     mail_address VARCHAR(50) NOT NULL,
62     department_id INT NOT NULL,
63     PRIMARY KEY (professor_id),
64     FOREIGN KEY (department_id) REFERENCES Departments(department_id),
65     UNIQUE (mail_address)
66 );
67
68 CREATE TABLE Classrooms
69 (
70     class_id INT NOT NULL,
71     class_name VARCHAR(50) NOT NULL,
72     class_building VARCHAR(50) NOT NULL,
73     course_id INT NOT NULL,
74     professor_id INT NOT NULL,
75     PRIMARY KEY (class_id),
76     FOREIGN KEY (course_id) REFERENCES Courses(course_id),
77     FOREIGN KEY (professor_id) REFERENCES Professors(professor_id)
78 );

```

בשאלתה זו אוו מעוניינים ליצור את כל הטבלאות של מבנה הנתונים שלנו, כאשר כל טבלה אנחנו מגדירים היטב את כל המאפיינים של אותה הטבלה. כמובן שחובה לוודא שיש לנו את כל המפתחות הראשיים והמפתחות הזרים בכל אחד מהטבלאות שנוצרו.

:Drop tables - SQL

```
dropTables.sql insertTables.sql selectAll.sql
1 DROP TABLE IF EXISTS Grades CASCADE;
2 DROP TABLE IF EXISTS Enrollments CASCADE;
3 DROP TABLE IF EXISTS classrooms CASCADE;
4 DROP TABLE IF EXISTS Professors CASCADE;
5 DROP TABLE IF EXISTS Courses CASCADE;
6 DROP TABLE IF EXISTS Departments CASCADE;
7 DROP TABLE IF EXISTS Students CASCADE;
```

בשאלת SQL זו אנחנו מוחקים טבלה מבסיס הנתונים שלנו, יחד עם כל הרשומות הקיימות באותה טבלה. לצורך העניין, אם קיימים לנו 400 סטודנטים בטבלת הסטודנטים אז לאחר הפעלת שאילתה זו הטבלה, וכל 400 הרשומות שהיו קיימות בה ימחקו לצמיתות.
* שימו לב שאין אפשרות לבטל פעולת מחיקה זו ולכן חשוב מאוד לדאוג שיהיה בכל עם גיבוי עדכני לכל הנתונים שלנו.

חשוב לדעת שכאשר מריצים את השאילתה היא עובדת על כל הטבלאות בפרויקט, כלומר היא תמחק את כל הנתונים שיש לנו בפרויקט. אם אנחנו מעוניינים למחוק טבלה ספציפית אז אנחנו צריכים לסמן את השורה של מחיקת הטבלה הספציפית הזאת ואז להריץ את השאילתה.

:Insert tables - SQL

```

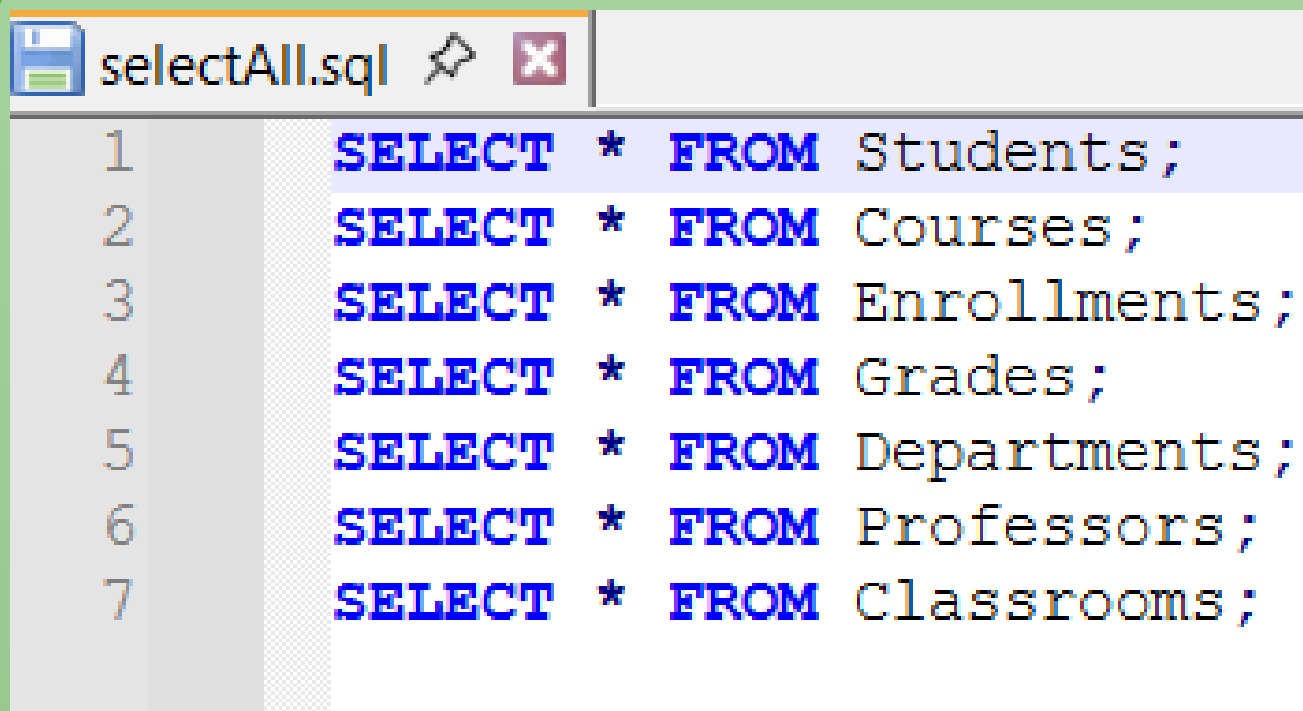
1  -- Students
2  INSERT INTO Students (student_id, first_name, last_name, birth_date, mail_address) VALUES
3  (1, 'Lior', 'Masas', '1998-11-17', 'lior.m.masas@gmail.com'),
4  (2, 'John', 'Doe', '2000-05-23', 'john.doe@gmail.com'),
5  (3, 'Jane', 'Smith', '1999-08-30', 'jane.smith@gmail.com');
6
7  -- Courses
8  INSERT INTO Courses (course_id, course_name, credits) VALUES
9  (101, 'Math 101', 3),
10 (102, 'History 101', 4),
11 (103, 'Computer Science 101', 3);
12
13 -- Enrollments
14 INSERT INTO Enrollments (enrollment_id, enrollment_date, enrollment_price, student_id, course_id) VALUES
15 (1, '2023-09-01', 500, 1, 101),
16 (2, '2023-09-01', 450, 2, 102),
17 (3, '2023-09-01', 600, 3, 103);
18
19 -- Grades
20 INSERT INTO Grades (grade, grade_date, pass_fail, student_id, course_id) VALUES
21 (1, 101, 90, '2023-12-01', 'pass'),
22 (2, 102, 85, '2023-12-01', 'pass'),
23 (3, 103, 78, '2023-12-01', 'fail');
24
25 -- Departments
26 INSERT INTO Departments (department_id, department_name, head_professor_id, department_building) VALUES
27 (1, 'Computer Science', 101, 'Building A'),
28 (2, 'Mathematics', 102, 'Building B'),
29 (3, 'History', 103, 'Building C');
30
31 -- Professors
32 INSERT INTO Professors (professor_id, first_name, last_name, hireing_date, wage, mail_address, department_id) VALUES
33 (101, 'David', 'Johnson', '2015-08-15', 70000, 'david.johnson@university.com', 1),
34 (102, 'Emily', 'Williams', '2017-03-10', 75000, 'emily.williams@university.com', 2),
35 (103, 'Michael', 'Brown', '2018-06-25', 72000, 'michael.brown@university.com', 3);
36
37 -- Classrooms (class)
38 INSERT INTO Classrooms (class_id, class_name, class_building, course_id, professor_id) VALUES
39 (1, 'CS101 - Lecture', 'Room 101', 103, 101),
40 (2, 'Math101 - Lecture', 'Room 102', 102, 102),
41 (3, 'History101 - Lecture', 'Room 103', 101, 103);
42

```

בשאלית זה אנחנו רוצים להכניס נתונים לטבלאות שלנו באמצעות שאילתת SQL ונתונים בפורמט של קובץ CSV.

יש מספר רשומות שאנחנו רוצים להכניס לטבלה כדי לבדוק שאכן הכנסת הנתונים עובדת כראוי. אחת מהטבלאות שלנו.

:Select all - SQL



```

1  SELECT * FROM Students;
2  SELECT * FROM Courses;
3  SELECT * FROM Enrollments;
4  SELECT * FROM Grades;
5  SELECT * FROM Departments;
6  SELECT * FROM Professors;
7  SELECT * FROM Classrooms;
  
```

באמצעות שאילתת SQL זו אנחנו נציג את כל הטבלאות הקיימות לנו, וכן את כל הרשומות הקיימות בכל אחת מהטבלאות.

כיוון שאי אפשר להציג את כל במקביל, ה - postgres יציג לנו את הטבלה האחרונה שרשומה בשאלתה (במקרה שלנו טבלת הכיתות). אם אנחנו רוצים להציג טבלה מסוימת יש לסמן את השורה של אותה הטבלה ואז להריץ את השאילתה.

הסבר עיצובי של מסד הנתונים:

המערכת מבוססת על עקרונות נורמליזציה שמחלקים את הנתונים לישויות נפרדות ומיצגים בצורה ברורה את הקשרים ביניהן. כל ישות (כמו סטודנט, קורס, מרצה וכו') שומרת על המידע שלה בצורה עצמאית, ובו בזמן כל קשר בין ישות אחת לשנייה מטופל בטבלאות מקשרות, מה שמבטיח גמישות ושלמות נתונים.

למשל, הקשרים בין סטודנטים לקורסים מנוהלים באמצעות טבלאות grades - i enrolments כאשר כל רישום וציונים נשמרים בנפרד ומכילים פרטי תאריך, עלות, ציונים ועוד. זה מספק את הגמישות להוסיף או לשנות נתונים בצורה מדויקת ומאפשר שליטה מלאה על תיעוד ההרשמות וההישגים.

בנוסף, ישנה היררכיה ברורה בין המחלקות השונות במוסד האקדמי. כל קורס שייך למחלקה אחת, ומרצים משתייכים למחלקות שבראשן עומד ראש מחלקה. הקשרים האלו מאפשרים ניהול משאבים בצורה גמישה ומאפשרים מעקב אחרי הסגל האקדמי ותחומי ההתמחות השונים.

הקשר בין קורסים, מרצים וכיתות הלימוד מנוהל באמצעות טבלאות classrooms, שמספקת מידע על מיקום ושם כל כיתה. זה מבטיח אפשרות לבניית מערכת שעות מסודרת ומאפשר תיאום בין המרצים והחדרים בצורה פשוטה ויעילה.

לבסוף, העיצוב כולו שומר על תקינות ושלמות הנתונים בעזרת שימוש במפתחות ראשיים (PRIMARY KEY) וזרים (FOREIGN KEY), ומונע חזרתיות של נתונים או זיהוי שגוי, כמו למשל הימנעות משמירה כפולה של כתובת מייל.

לסיכום, העיצוב לא רק שמספק פתרון מלא ונכון לוגית להיום, אלא גם מאפשר הרחבה עתידית של המערכת בקלות, כך שניתן להוסיף תכנים כמו נוכחות, שעות עבודה, או אפילו אינטגרציה עם מערכות חיצוניות.

שיטות הכנסת הנתונים שנבחרו:

ו. יצירת סקריפט בפייתון:

```

1 # Description: This script generates CSV files for each table in the database.
2 > import ...
3
4
5
6 fake = Faker()
7
8 # create a CSV file for students table
9 def generate_students(filename, num_rows=400): 1usage
10     with open(filename, mode='w', newline='') as file:
11         writer = csv.writer(file)
12         writer.writerow(["student_id", "first_name", "last_name", "birth_date", "mail_address"])
13         for i in range(1, num_rows + 1):
14             writer.writerow([i, fake.first_name(), fake.last_name(), fake.date_of_birth(minimum_age=18, maximum_age=30), fake.email()])
15
16 # create a CSV file for courses table
17 def generate_courses(filename, num_rows=100): 1usage
18     with open(filename, mode='w', newline='') as file:
19         writer = csv.writer(file)
20         writer.writerow(["course_id", "course_name", "credits"])
21         for i in range(1, num_rows + 1):
22             writer.writerow([i, fake.sentence(nb_words=3).replace(_old: '.', _new: ''), random.randint(a: 1, b: 5)])
23
24 # create a CSV file for enrollments table
25 def generate_enrollments(filename, num_rows=400): 1usage
26     with open(filename, mode='w', newline='') as file:
27         writer = csv.writer(file)
28         writer.writerow(["enrollment_id", "enrollment_date", "enrollment_price", "student_id", "course_id"])
29         for i in range(1, num_rows + 1):
30             writer.writerow([i, fake.date_this_decade(), random.randint(a: 500, b: 5000), random.randint(a: 1, b: 400), random.randint(
31
32 # create a CSV file for grades table
33 def generate_grades(filename, num_rows=400): 1usage
34     with open(filename, mode='w', newline='') as file:

```

שיטות הכנסת הנתונים שנהגו:

2. mockaroo.

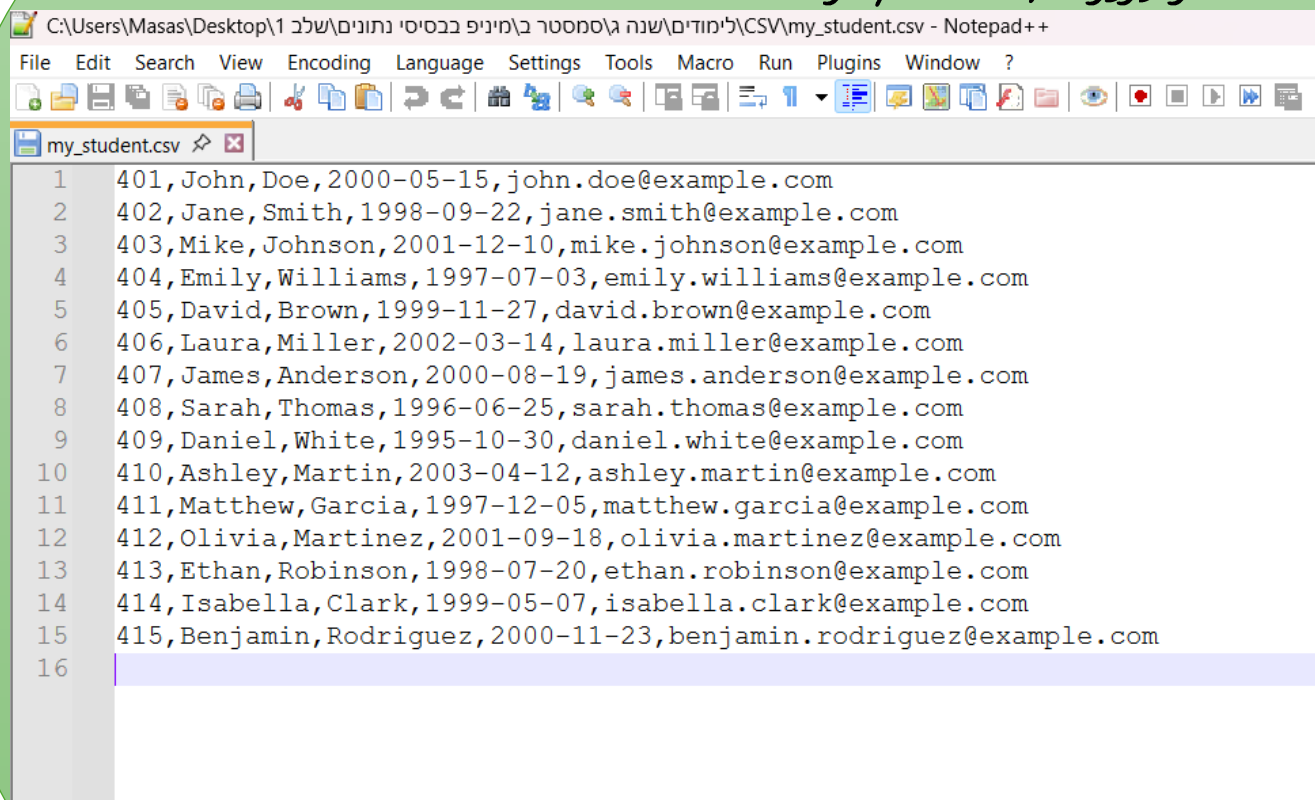
| Field Name | Type | Options |
|--------------|---------------|---|
| student_id | Row Number | blank: 0 % Σ X |
| first_name | First Name | blank: 0 % Σ X |
| last_name | Last Name | blank: 0 % Σ X |
| birth_date | Datetime | 01/01/1970 to 03/31/2005 format: dd/mm/yyyy blank: 0 % Σ X |
| mail_address | Email Address | blank: 0 % Σ X |

+ ADD ANOTHER FIELD GENERATE FIELDS USING AI...

Rows: 400 Format: CSV Line Ending: Windows (CRLF) Include: ☒ header ☐ BOM

שיטות הכנסת התעונים שנבחרו:

3. הכנסת תעונים מקבצים באופן ידני:



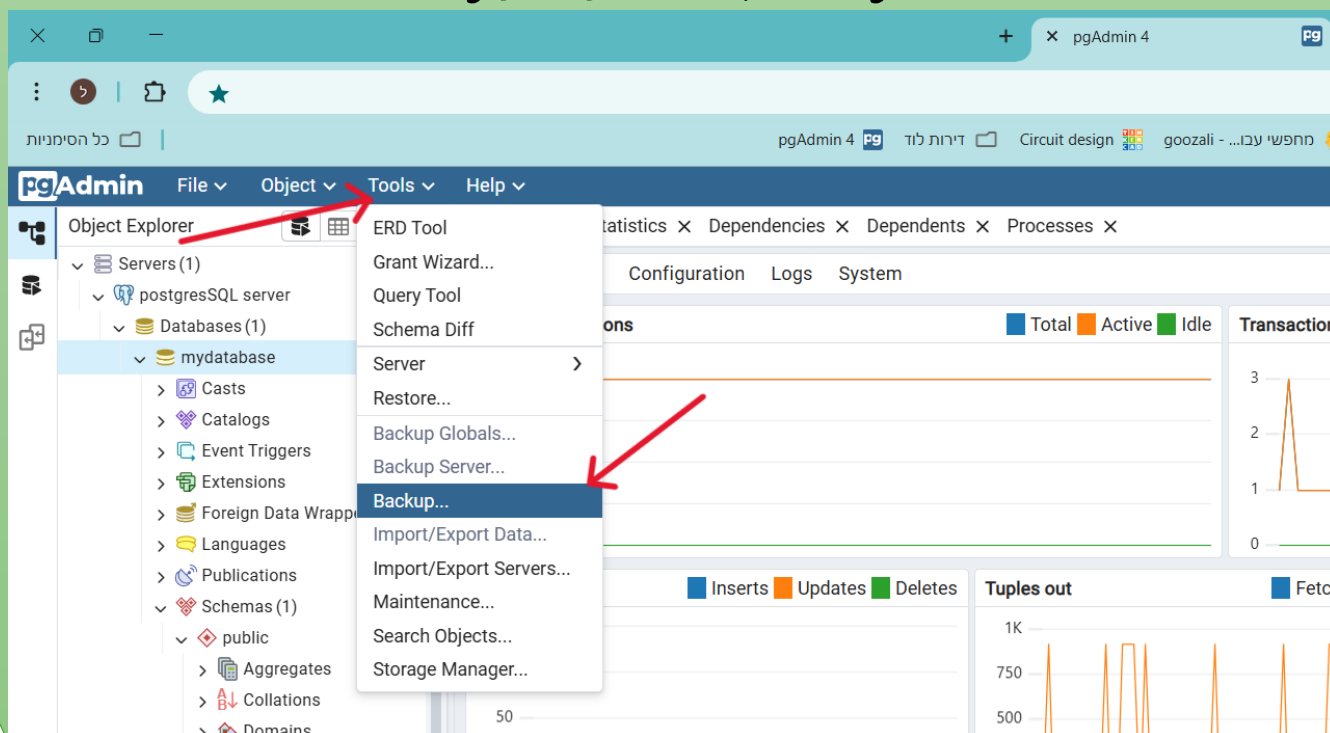
```

C:\Users\Masas\Desktop\1\שלב\נתונים\בסיסי\CSV\my_student.csv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
my_student.csv
1 401, John, Doe, 2000-05-15, john.doe@example.com
2 402, Jane, Smith, 1998-09-22, jane.smith@example.com
3 403, Mike, Johnson, 2001-12-10, mike.johnson@example.com
4 404, Emily, Williams, 1997-07-03, emily.williams@example.com
5 405, David, Brown, 1999-11-27, david.brown@example.com
6 406, Laura, Miller, 2002-03-14, laura.miller@example.com
7 407, James, Anderson, 2000-08-19, james.anderson@example.com
8 408, Sarah, Thomas, 1996-06-25, sarah.thomas@example.com
9 409, Daniel, White, 1995-10-30, daniel.white@example.com
10 410, Ashley, Martin, 2003-04-12, ashley.martin@example.com
11 411, Matthew, Garcia, 1997-12-05, matthew.garcia@example.com
12 412, Olivia, Martinez, 2001-09-18, olivia.martinez@example.com
13 413, Ethan, Robinson, 1998-07-20, ethan.robinson@example.com
14 414, Isabella, Clark, 1999-05-07, isabella.clark@example.com
15 415, Benjamin, Rodriguez, 2000-11-23, benjamin.rodriguez@example.com
16

```

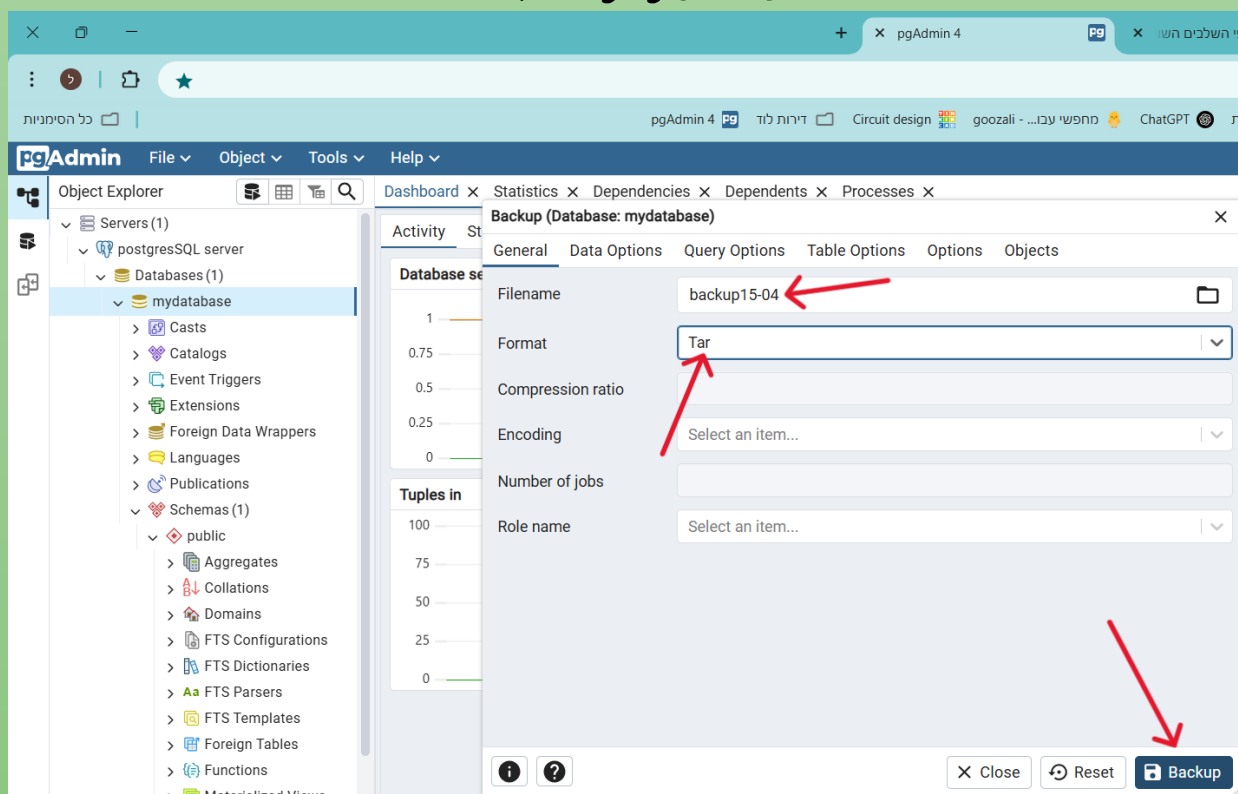
גיבוי התנאים:

בשלב הראשון של גיבוי התנאים, נבחר בסרגל הכלים העליון של postgres באפשרות tools ואחר כך נבחר ב- Backup.. כמתואר בתמונה:



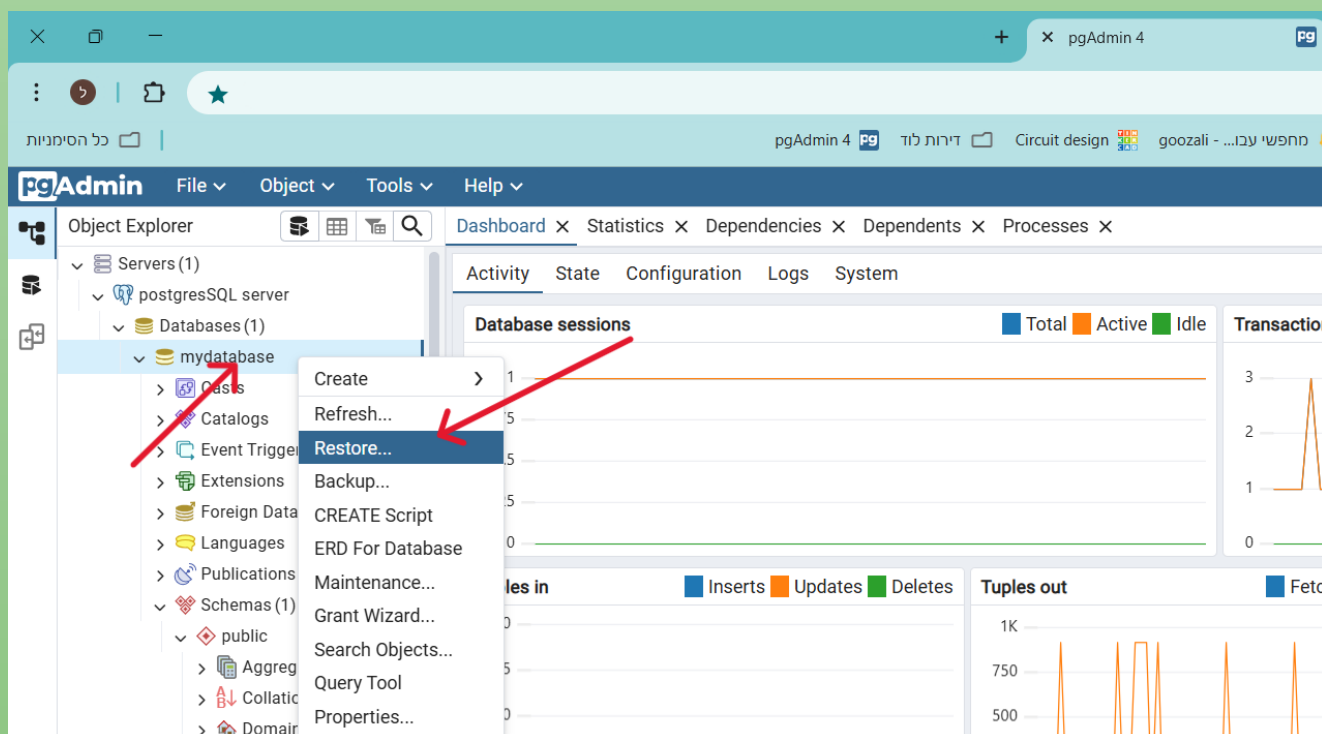
גיבוי התנאים:

לאחר מכן, נבחר שם לגיבוי, בדרך כלל זה יהיה backup עם תאריך הגיבוי. נצייר את פורמט הקובץ כמתואר בתמונה ונחץ על backup.



שחזור התענים:

כדי לשחזר תענים מגיבוי שעשינו, יש לאחזר מקש ימני על בסיס התענים שאליה אנחנו רוצים לשחזר את התענים, ולאחר מכן לאחזר על כפתור Restore..



שחזור התנאים:

לאחר מכן, נבחר את הפורמט המתאים, נבחר את קובץ הגיבוי מה- storage manager, ולבסוף נאשר את הבחירות בכפתור Restore.

*יכול להיות שלא נראה את השינויים מיד, ונצטרך לרענן את הטבלאות שלנו כדי לראות את העדכון.

