בס"ד

כותבי הדוח:

אריאל פאליק - 065974289

רועי בוק - 208560102

**שלב 1)**

הארגון שלנו – אוניברסיטה / מכללה / סמינר.

הארגון שלנו כולל כמה וכמה תת אגפים כמו מרצים, סטודנטים, מעונות, מבחנים **וציונים** (שזה האגף שאנחנו בחרנו- ציונים).

1. סטודנטים
2. מרצים
3. מבחנים
4. ציונים
5. קורסים
6. סמסטר

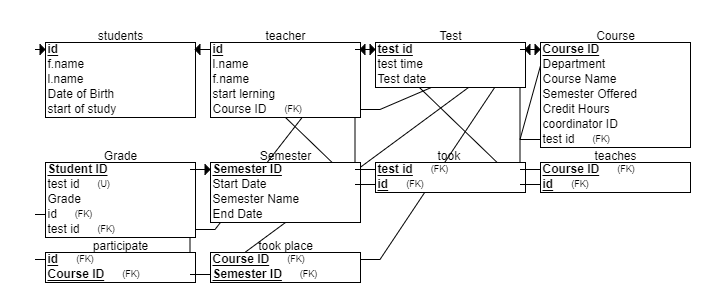
מטרת המערכת היא לקבל נתונים על הציונים של הסטודנטים ולנתח אותם בשיטות סטטיסטיות על מנת להעלות את ממוצע הציונים.

תרשים ERD:

תמונה שמכילה תרשים, ציור, שרטוט, לבן

התיאור נוצר באופן אוטומטי

תרשים DSD:



**נרמול הסכימה:**

הסכימה נמצאת ברמת נרמול NF-,3 מכיוון שלא קיימות תכונות מורכבות ,(1-NF)אין תכונות התלויות רק בחלק מהמפתח ,(2-NF) ואין תכונות התלויות בתכונות אחרות שאינן חלק המפתח(3-NF) .

CREATE TABLE students (

id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

date\_of\_birth DATE,

start\_of\_study INT

);

CREATE TABLE teacher (

id INT PRIMARY KEY,

fname VARCHAR(50),

lname VARCHAR(50),

start\_learning INT

);

CREATE TABLE course (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(100),

department VARCHAR(100),

credit\_hours INT,

coordinator\_id INT,

FOREIGN KEY (coordinator\_id) REFERENCES teacher(id)

);

CREATE TABLE test (

test\_id INT PRIMARY KEY,

test\_time TIME,

test\_date DATE,

course\_id INT,

FOREIGN KEY (course\_id) REFERENCES course(course\_id)

);

CREATE TABLE grade (

student\_id INT,

test\_id INT,

grade INT,

PRIMARY KEY (student\_id, test\_id),

FOREIGN KEY (student\_id) REFERENCES students(id),

FOREIGN KEY (test\_id) REFERENCES test(test\_id)

);

CREATE TABLE semester (

semester\_id INT PRIMARY KEY,

semester\_name VARCHAR(50),

start\_date DATE,

end\_date DATE

);

CREATE TABLE course\_teacher (

course\_id INT,

teacher\_id INT,

PRIMARY KEY (course\_id, teacher\_id),

FOREIGN KEY (course\_id) REFERENCES course(course\_id),

FOREIGN KEY (teacher\_id) REFERENCES teacher(id)

);

CREATE TABLE student\_test\_participation (

student\_id INT,

test\_id INT,

PRIMARY KEY (student\_id, test\_id),

FOREIGN KEY (student\_id) REFERENCES students(id),

FOREIGN KEY (test\_id) REFERENCES test(test\_id)

);

CREATE TABLE course\_semester (

course\_id INT,

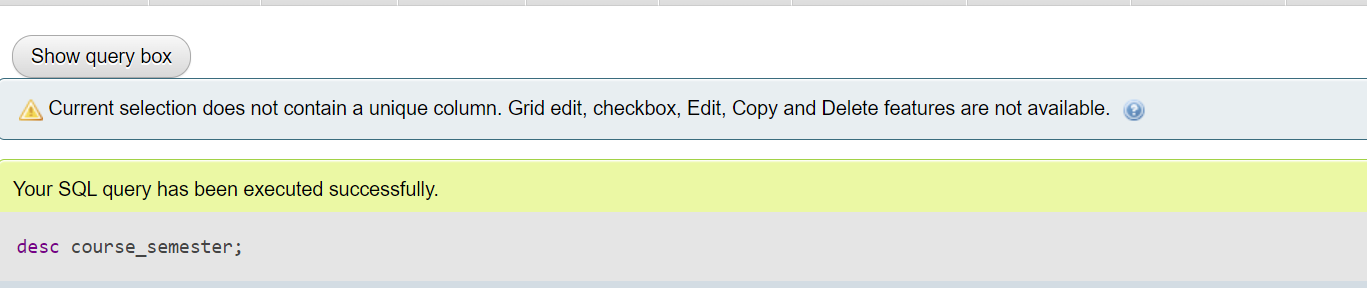
semester\_id INT,

PRIMARY KEY (course\_id, semester\_id),

FOREIGN KEY (course\_id) REFERENCES course(course\_id),

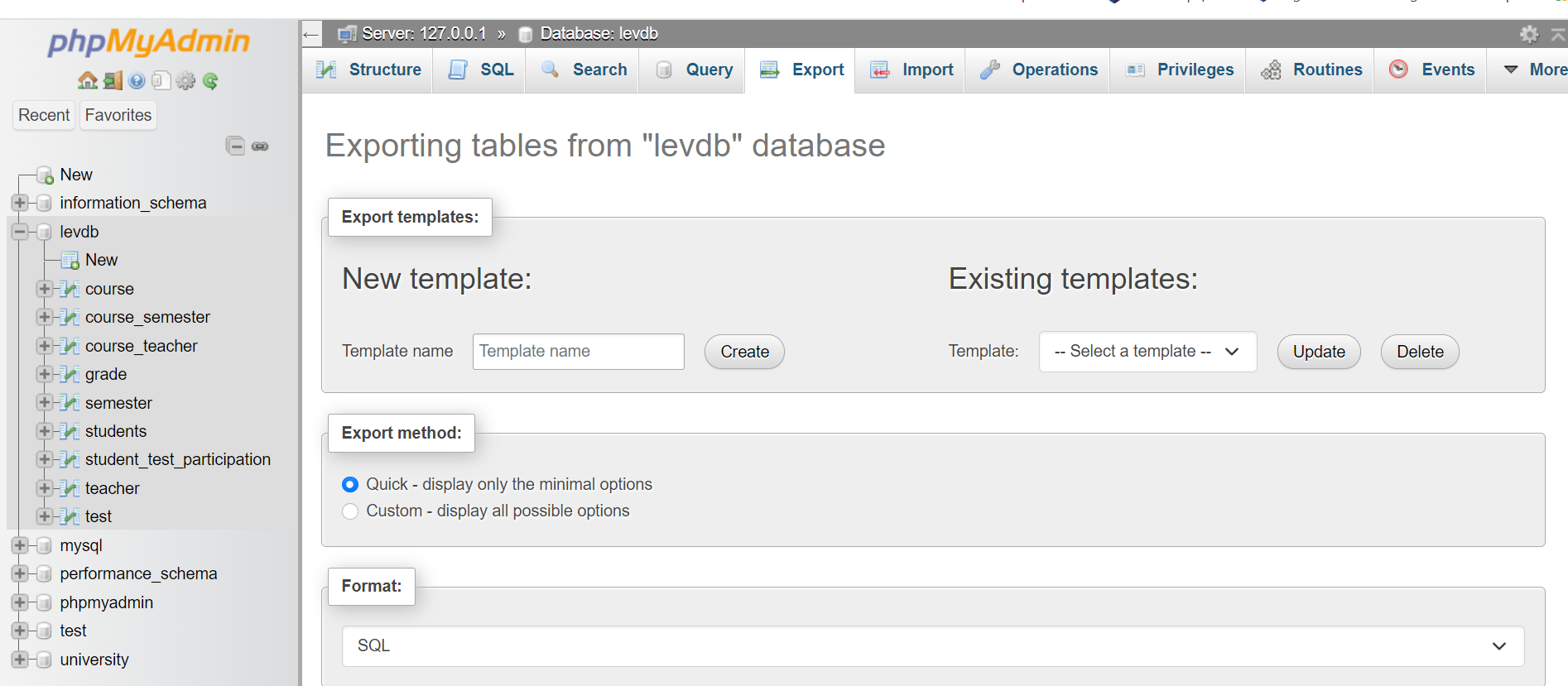
FOREIGN KEY (semester\_id) REFERENCES semester(semester\_id)

);



תמונה שמכילה טקסט, תוכנה, גופן, סמל מחשב

התיאור נוצר באופן אוטומטי



ולהלן שיטת ההכנסה ע"פ פשט:

תמונה שמכילה טקסט, תוכנה, מספר, צילום מסך

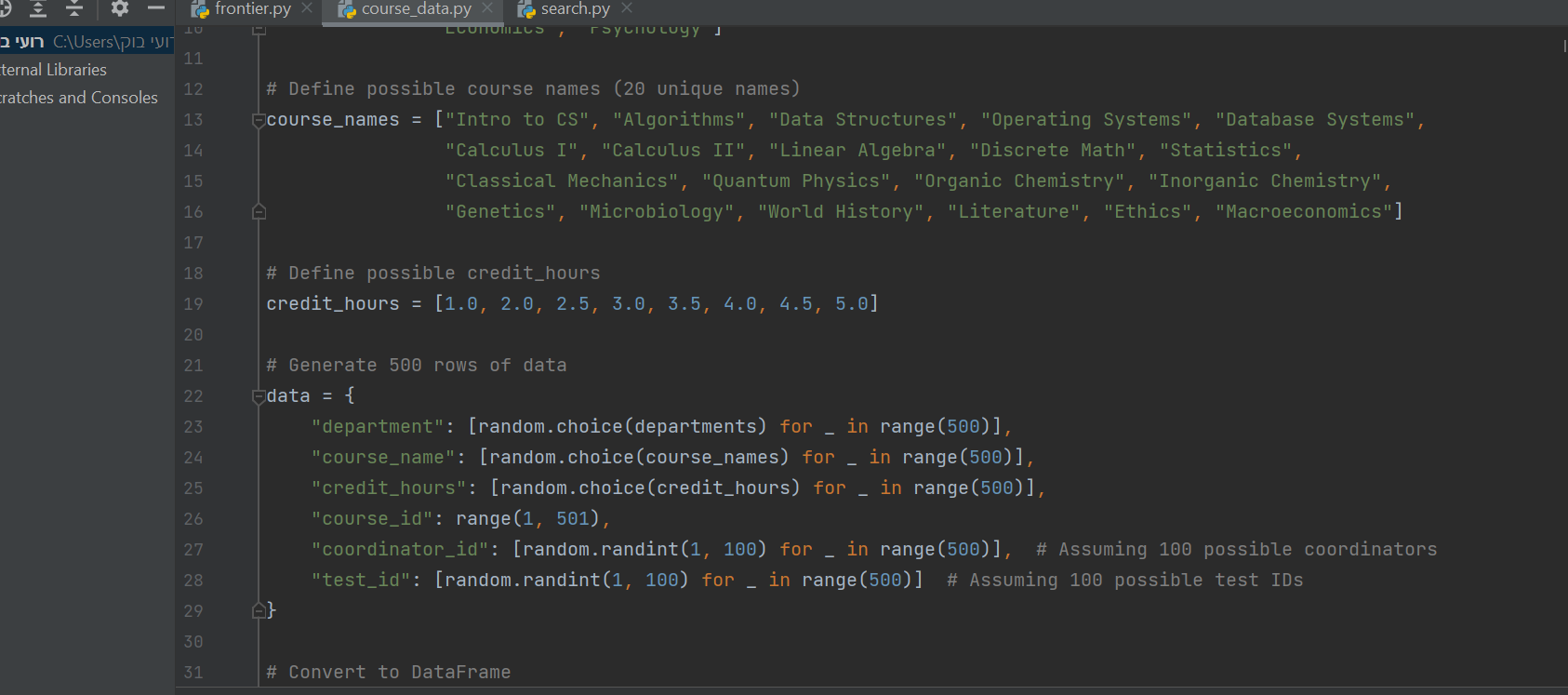
התיאור נוצר באופן אוטומטי

וכן עוד שיטה של הכנסה ע"פ עוד פשט:

תמונה שמכילה טקסט, תוכנה, צילום מסך, מספר

התיאור נוצר באופן אוטומטי

וכן עוד שיטה ע"פ רבני הפייתון:



וכן המקום להודיע להודיע להודיע (שכל קוויך וכו') שלא השתמשנו בפלט של הקבצי פייתון מכיוון שלא ראינו צורך בטרחה מרובה (שאסורה בימים אלו כמובא בספרים הק'..) ולכן תהנו מיצירת המופת שלנו (סליחה בעצם של GPT)

גלוי וידוע לפנינו שאמורה לצאת טבלה אך מפני ענוותנותה של המערכת היא רק דיווחה כי "כל כבודה בת מלך פנימה"

מכיוון שמרצנו הק' ציווה עלינו לעשות את הפרוייקט בmysql וphpmyadmin שלא כשאר המרצים שליט"א והמרצות תליט"א כך יוצא מכל הנ"ל שההוראות במודל הם לפי הרוב כי "יחיד ורבים הלכה כרבים"

לכן דוח זה נראה כאילו הוא לאחר תעניות רבות וסיגופים אך אל לכם להביט בקנקן אלא לשתות את היין ההונגרי שנאמר "טעמו וראו כי טוב"

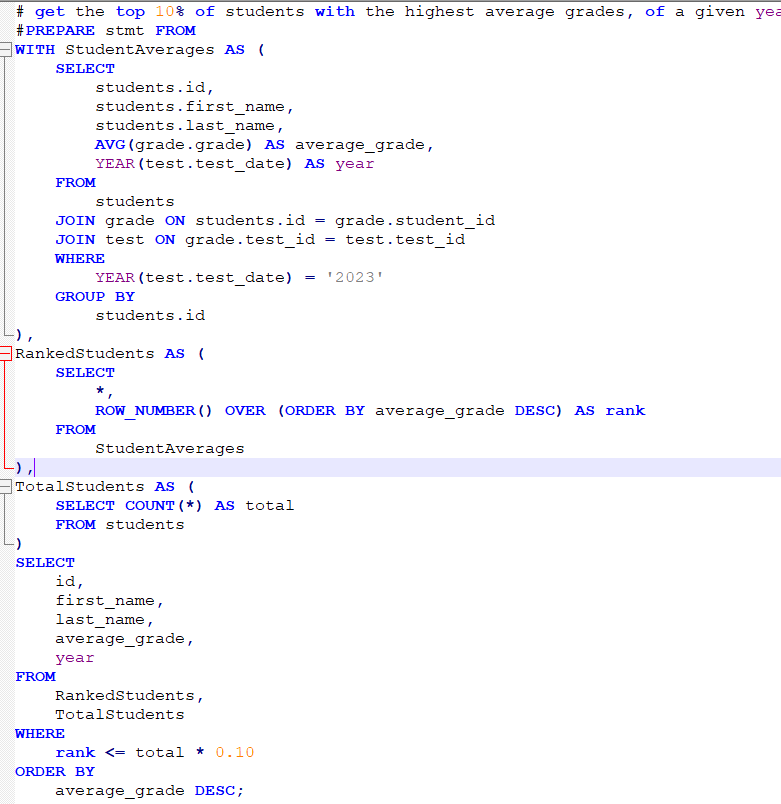
זיי גזעונט ובשורות טובות ישועות ונחמות ובבניין ירושלים ננוחם.

נ.ב. גם אם הציון לא משו לפחות נא להוסיף ציון על הפקטור...

**שלב ב**

להלן נציג את השאילתות שלנו ומה רצינו לעשות וכו' וכו'

השאילתה הראשונה שלנו היא: המטרה שלנו היא למצוא את 10% של הסטודנטים בעלי הציון הגבוה בשנה מסויימת (בעולם האמיתי האלה שיזכו בהצטיינות בסוף שנה..)



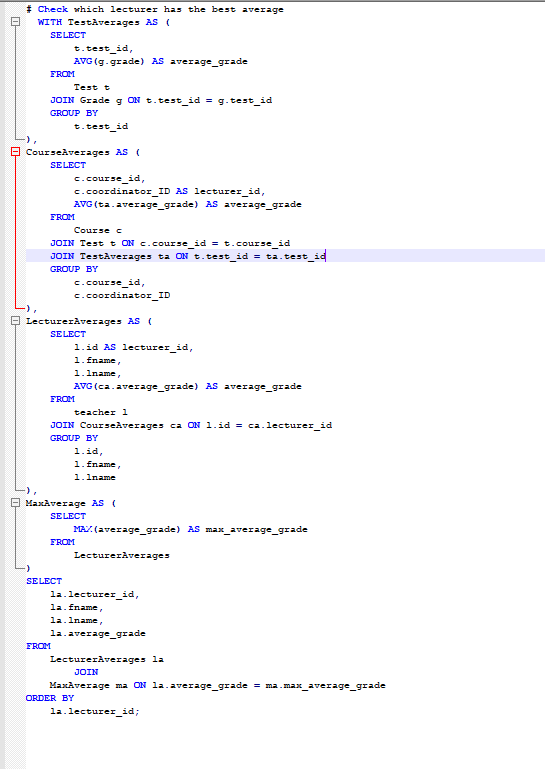
תוצאות השאילתה:

תמונה שמכילה טקסט, צילום מסך, גופן, מספר

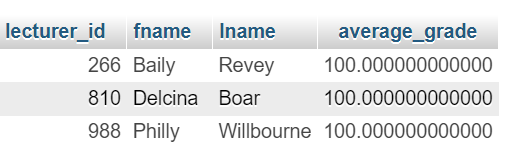
התיאור נוצר באופן אוטומטי

השאילתה השנייה שלנו:

באה לגלות מי המרצה שתלמידיו הגיעו להישגים הכי טובים (אולי יש יותר מ1..)



תוצאות השאילתה:



השאילתה השלישית שלנו: באה לבדוק איזה קורסים היו בעלי ציונים הכי גבוהים מבין הקורסים בשנה מסויימת (שימוש בdate..)

תמונה שמכילה טקסט, צילום מסך, תוכנה

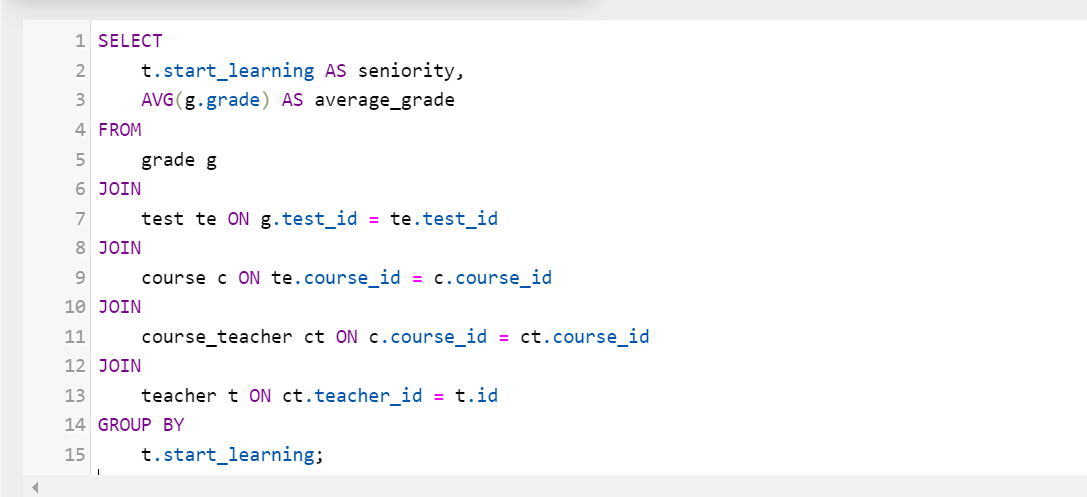
התיאור נוצר באופן אוטומטי

והנה התוצאות של השאילתה:

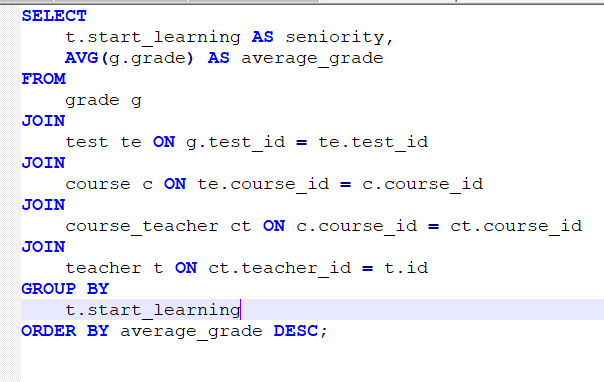
תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

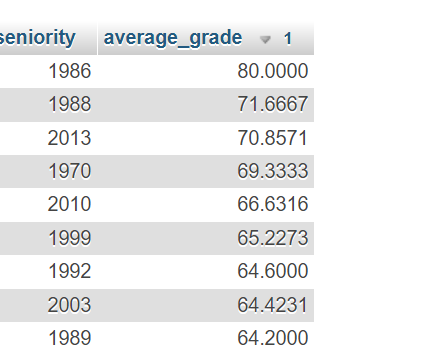
בשאילתה הרביעית

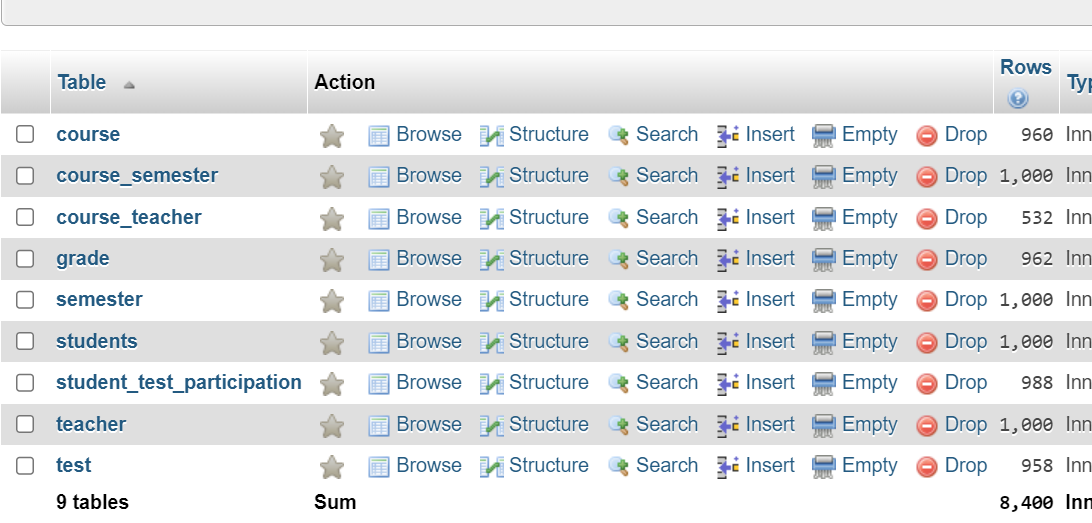


באה לבדוק השפעת הוותק של המרצה על הציונים אך לאחר הרצות גילינו שקיים טעות בבסיס נתונים (שהמוק דאטה ייצר לנו כל המרצים לימדו את אותו קורס...) ולכן שינינו את הבסיס נתונים ולהן השאילתה המתוקנת:

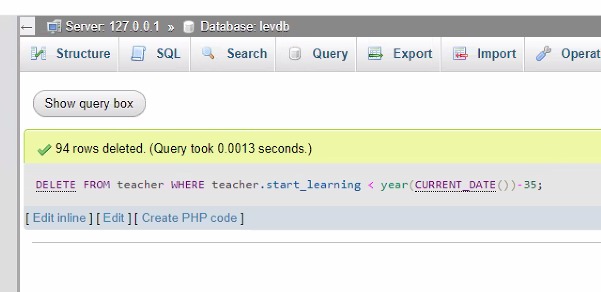


והנה התוצאות של השאילתה:





להלן שאילתות המחיקה שלנו: השאילתה הראשונה בעצם באה למחוק נתונים על מרצים שיצאו לפנסיה

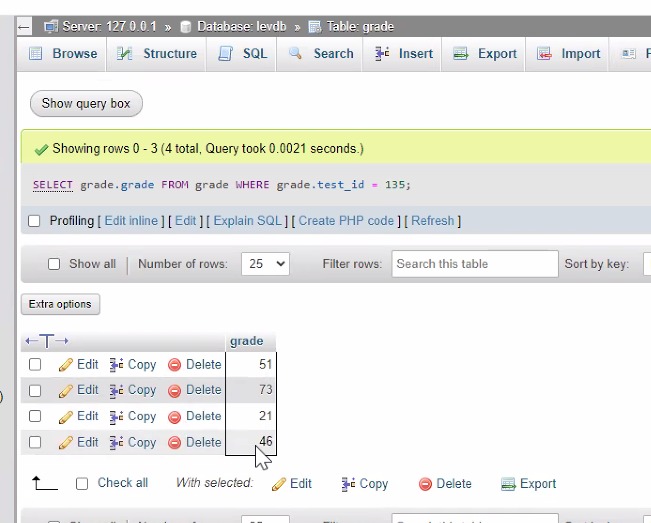




תמונה שמכילה טקסט, צילום מסך, תוכנה, דף אינטרנט

התיאור נוצר באופן אוטומטיהשאילתה השניה באה למחוק את כל המרצים שבעצם לא לימדו אף קורס (הבסיס נתונים שלנו היה טוב ולכן לא היו כאלה..)

שאילתות עדכון:

לפני:   
 הקוד: תמונה שמכילה טקסט, גופן, מספר, קו

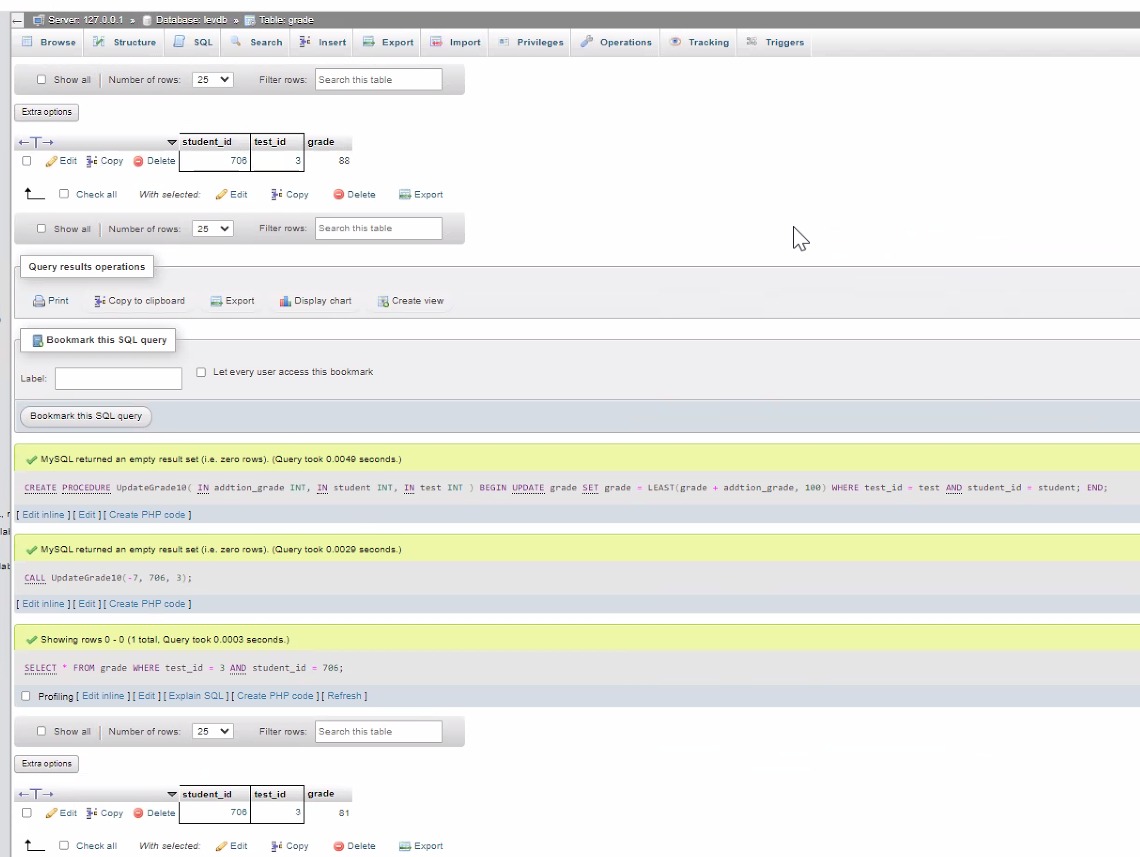
התיאור נוצר באופן אוטומטי

אחרי:

תמונה שמכילה טקסט, צילום מסך, תוכנה, סמל מחשב

התיאור נוצר באופן אוטומטי  
השאילתה מדמה פקטור של 10 אחוז החל מציון מסויים (דבר שישנו במכון..)

עוד שאילתת עדכון שעשינו היא באה לדמות ערעור של סטודנט על המבחן (יש יכולת שהציון יעלה אך גם שירד כמו בדוגמא שלנו וכבר היו דוגמאות במכון...) ניתן לראות את הטבלה לפני, את הקוד ואת הטבלה לאחר השינוי..

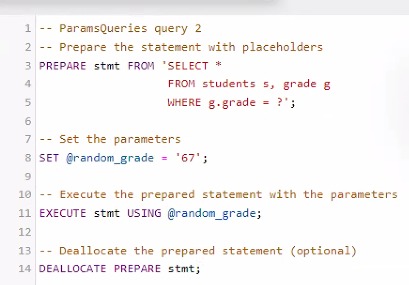


שאילתה עם פרמטרים:

תמונה שמכילה טקסט, צילום מסך, גופן

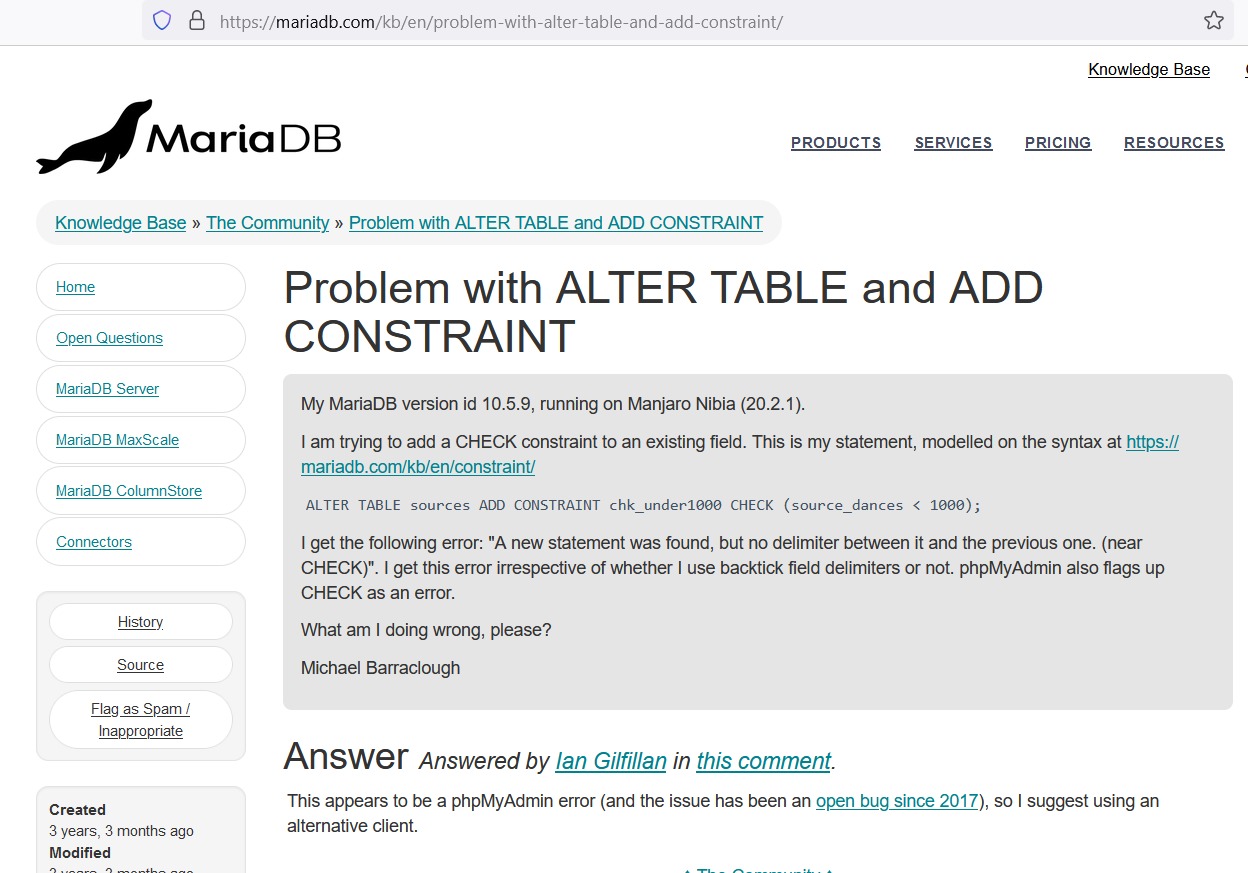
התיאור נוצר באופן אוטומטיבעצם בדקנו ממוצע של פרק זמן מסויים (יעיל לבדוק האם בזמן קורונה או מלחמה הממוצע נמוך מאשר סתם זמן מקביל וכו..)

עוד שאילתה עם פרמטרים זה בא להקל על מזכירות ציונים (בעצם כל ציון שתכניס נקבל רשימה של הסטודנטים שקיבלו אותו)



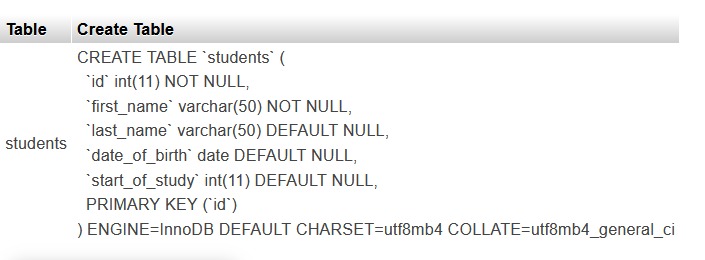
ולהלן האילוצים שעשינו ותיאור וכו' אמממה בעקבות ניסיונותינו לעשות אילוצים ועקב שגיאות חוזרות ונשנות

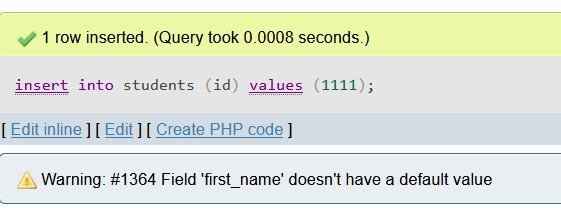
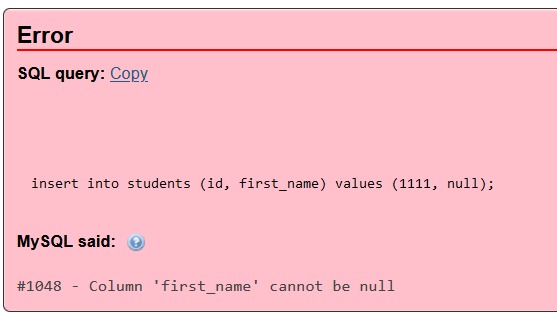
הרינו מודיעים קבל עם ועדה (איזה עדה שתבחרו) שיש בעייה לעשות זאת והנה ההוכחה  
מ.ש.ל.



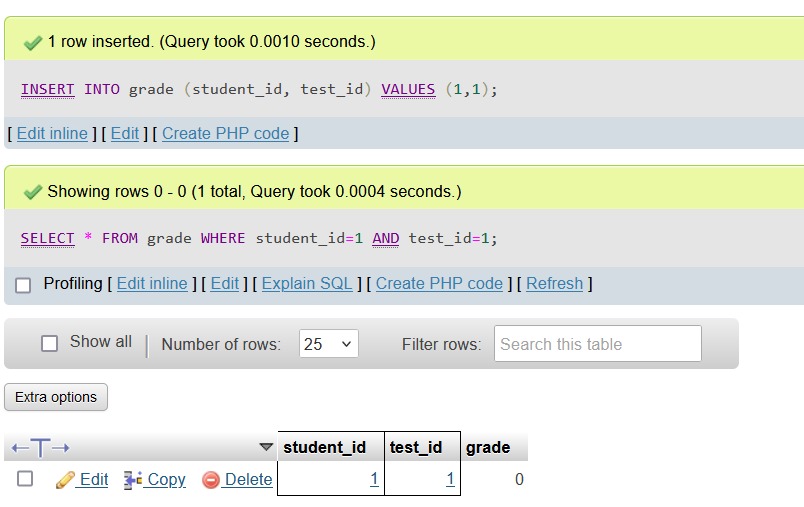
אילוץ שכן עבד לנו זה  
alter table students modify first\_name varchar(50) NOT NULL;

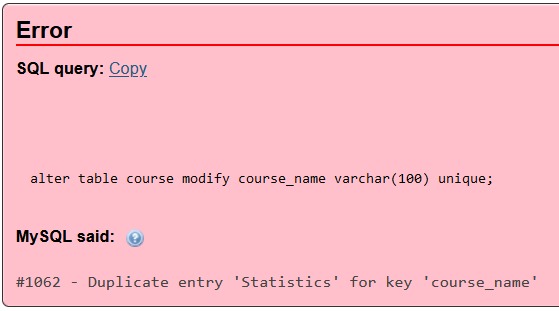
האילוץ אומר לנו שחייב שיהיה שם פרטי לסטודנט





עוד אילוץ alter table grade modify grade int(11) default 0;  
בעצם כל ציון שאין לו ציון יקבל ציון 0



עוד אילוץ ששם הקורס יהיה ייחודי alter table course modify course\_name varchar(100) unique;  
  


**~~~שלב ג)~~~**

**2 פונקציות:**

פונקציה של קורסים:

פונקציה זו מחזירה את הקורסים שהסטודנט נבחן בהם, והרי הקוד שלה (קודה):

DELIMITER //

CREATE FUNCTION get\_student\_courses(p\_student\_id INT) RETURNS VARCHAR(1000)

BEGIN

    DECLARE course\_list VARCHAR(1000) DEFAULT '';

    DECLARE course\_name VARCHAR(100);

    DECLARE done INT DEFAULT 0;

    DECLARE course\_cursor CURSOR FOR

        SELECT c.course\_name

        FROM course c

        JOIN student\_test\_participation stp ON c.course\_id = stp.test\_id

        WHERE stp.student\_id = p\_student\_id;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN course\_cursor;

    course\_loop: LOOP

        FETCH course\_cursor INTO course\_name;

        IF done THEN

            LEAVE course\_loop;

        END IF;

        SET course\_list = CONCAT(course\_list, course\_name, ', ');

    END LOOP;

    CLOSE course\_cursor;

    -- Remove the trailing comma and space

    IF CHAR\_LENGTH(course\_list) > 2 THEN

        SET course\_list = LEFT(course\_list, CHAR\_LENGTH(course\_list) - 2);

    END IF;

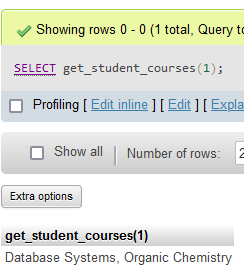
    RETURN course\_list;

END;

//

DELIMITER ;

הא לכם הוכחה שהפונקציה עובדת:



פונקציה של ציונים:

זוהי פונקציה המחזירה את ממוצע ציוניו של סטודנט, והרי קודה:

DELIMITER //

CREATE FUNCTION calculate\_average\_grade(p\_student\_id INT) RETURNS FLOAT

BEGIN

    DECLARE avg\_grade FLOAT;

    DECLARE total\_grades INT;

    DECLARE grade\_count INT;

    -- Handle exceptions

    DECLARE EXIT HANDLER FOR SQLEXCEPTION

    BEGIN

        -- Return -1 in case of error

        SET avg\_grade = -1;

    END;

    -- Calculate total grades and count

    SELECT SUM(grade), COUNT(\*) INTO total\_grades, grade\_count

    FROM grade

    WHERE student\_id = p\_student\_id;

    -- Calculate average

    IF grade\_count > 0 THEN

        SET avg\_grade = total\_grades / grade\_count;

    ELSE

        SET avg\_grade = 0;

    END IF;

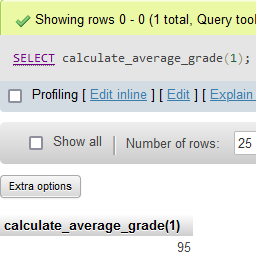
    RETURN avg\_grade;

END;

//

DELIMITER ;

הא לכם הוכחה שהפונקציה עושה את המוטל עליה:



**2 פרוצדורות:**

פרוצדורה של קורסים:

תפקידה של פרוצדורה זו הוא לשייך מרצה לקורס, והנה קודה:

DELIMITER //

CREATE PROCEDURE assign\_teacher\_to\_course(IN p\_course\_id INT, IN p\_teacher\_id INT)

BEGIN

    DECLARE course\_exists INT;

    DECLARE teacher\_exists INT;

    DECLARE error\_message VARCHAR(255) DEFAULT '';

    -- Check if the course exists

    SELECT COUNT(\*) INTO course\_exists FROM course WHERE course\_id = p\_course\_id;

    IF course\_exists = 0 THEN

        SET error\_message = 'Course does not exist';

        SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = error\_message;

    END IF;

    -- Check if the teacher exists

    SELECT COUNT(\*) INTO teacher\_exists FROM teacher WHERE id = p\_teacher\_id;

    IF teacher\_exists = 0 THEN

        SET error\_message = 'Teacher does not exist';

        SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = error\_message;

    END IF;

    -- Check if the teacher is already assigned to the course

    SELECT COUNT(\*) INTO teacher\_exists FROM course\_teacher WHERE course\_id = p\_course\_id AND teacher\_id = p\_teacher\_id;

    IF teacher\_exists > 0 THEN

        SET error\_message = 'Teacher is already assigned to the course';

        SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = error\_message;

    END IF;

    -- Attempt to assign teacher to course

    BEGIN

        DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

        BEGIN

            GET DIAGNOSTICS CONDITION 1

            error\_message = MESSAGE\_TEXT;

            SELECT CONCAT('Error assigning teacher to course: ', error\_message) AS detailed\_error;

        END;

        INSERT INTO course\_teacher (course\_id, teacher\_id) VALUES (p\_course\_id, p\_teacher\_id);

    END;

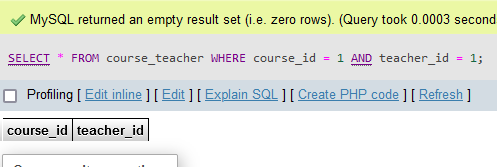
END;

//

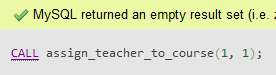
DELIMITER ;

רוצים לראות שהיא עובדת? קבלו:

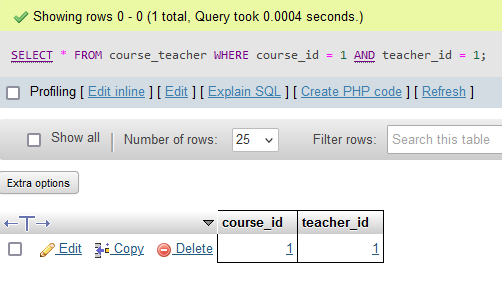
נבדוק, ברשותכם, האם המרצה מס' 1 מלמד את הקורס מס' 1:



חזיז ורעם! הוא לא מלמד את הקורס! נפנה מיד לתיקון העוול ההיסטורי (בתקוה שבג"ץ לא יפסול):



בחיל וברעדה נבדוק האם המרצה שובץ:



חי זקני! הלם ותדהמה! הפרוצדורה עבדה!

פרוצדורה של ציונים:

פרוצדורה זו מעדכנת את ציוניו של סטודנט, וזה לך הקוד:

DELIMITER //

CREATE PROCEDURE update\_student\_grade(IN p\_student\_id INT, IN p\_test\_id INT, IN p\_new\_grade INT)

BEGIN

    DECLARE old\_grade INT;

    -- Handle exceptions

    DECLARE EXIT HANDLER FOR SQLEXCEPTION

    BEGIN

        -- Output error message

        SELECT 'Error updating grade';

    END;

    -- Update grade

    SELECT grade INTO old\_grade FROM grade WHERE student\_id = p\_student\_id AND test\_id = p\_test\_id;

    IF old\_grade IS NOT NULL THEN

        UPDATE grade SET grade = p\_new\_grade WHERE student\_id = p\_student\_id AND test\_id = p\_test\_id;

    ELSE

        INSERT INTO grade (student\_id, test\_id, grade) VALUES (p\_student\_id, p\_test\_id, p\_new\_grade);

    END IF;

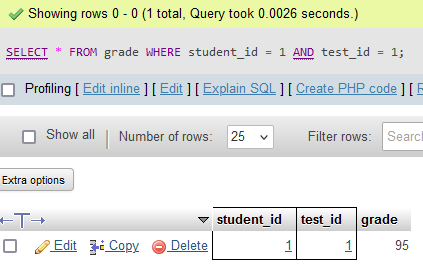
END;

//

DELIMITER ;

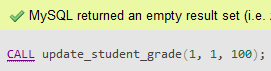
עובדת? יצאנו לבדוק:

זהו ציונו של הסטודנט מס' 1 בקורס מס' 1:

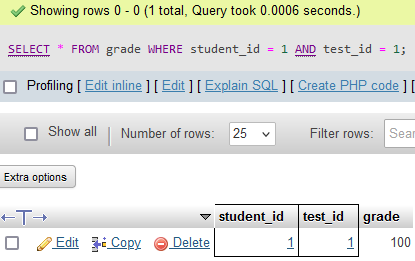


אכן כן, 95, אתם לא טועים. מצטיין נשיא.

הסטודנט שלנו לא פראייר ורוצה להתקבל למשרה הנחשקת של בודק תרגילים במכון לב, ולכן הוא מגיש ערעור, לאחר דין ודברים וחילופי אש המרצה נכנע, והמזת"ל ניגשים לעדכן את הציון:



המתח גואה, תופים בבקשה:



שיואו! עכשיו הוא יוכל לבדוק תרגילים בשכר מינימום!

**2 תוכניות ראשיות:**

תוכנית ראשית של קורסים:

תכנית זו, נו איך לומר, קוראת לפונקציה ולפרוצדורה של הקורסים. וכך היא נראית:

DELIMITER //

CREATE PROCEDURE main\_for\_courses()

BEGIN

    DECLARE student\_id INT DEFAULT 1;

    DECLARE course\_id INT DEFAULT 1;

    DECLARE teacher\_id INT DEFAULT 1;

    DECLARE courses VARCHAR(1000);

    -- Call procedure to assign teacher to course

    CALL assign\_teacher\_to\_course(course\_id, teacher\_id);

    -- Call function to get student's courses

    SET courses = get\_student\_courses(student\_id);

    -- Output the list of courses

    SELECT courses;

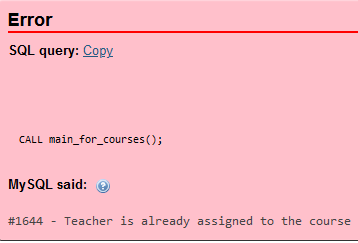
    -- the DECLARE and the SET are just to put in more programmatic stuff

END;

//

DELIMITER ;

התוכנית אמורה לזרוק הפעם חריגה, כי כבר שיבצנו את המרצה מס' 1 לקורס מס' 1, כזכור לקורא, ועל הדרך להדגים את הדפסת החריגות:



מהמם.

תוכנית ראשית של ציונים:

תכנית זאת מה עושה? קוראת לפונקציה ולפרוצדורה של הציונים, זה מה.

וזה הקוד:

DELIMITER //

CREATE PROCEDURE main\_for\_grades()

BEGIN

    DECLARE student\_id INT DEFAULT 1;

    DECLARE test\_id INT DEFAULT 1;

    DECLARE new\_grade INT DEFAULT 95;

    DECLARE avg\_grade FLOAT;

    -- Call procedure to update grade

    CALL update\_student\_grade(student\_id, test\_id, new\_grade);

    -- Call function to calculate average grade

    SET avg\_grade = calculate\_average\_grade(student\_id);

    -- Output the average grade

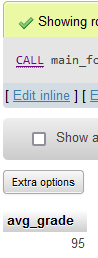
    SELECT avg\_grade;

END;

//

DELIMITER ;

נריץ:



וואו.

ובזאת ידידי, תמה לה הצגת שלב 3 המושלם, שבח לאל בורא עולם

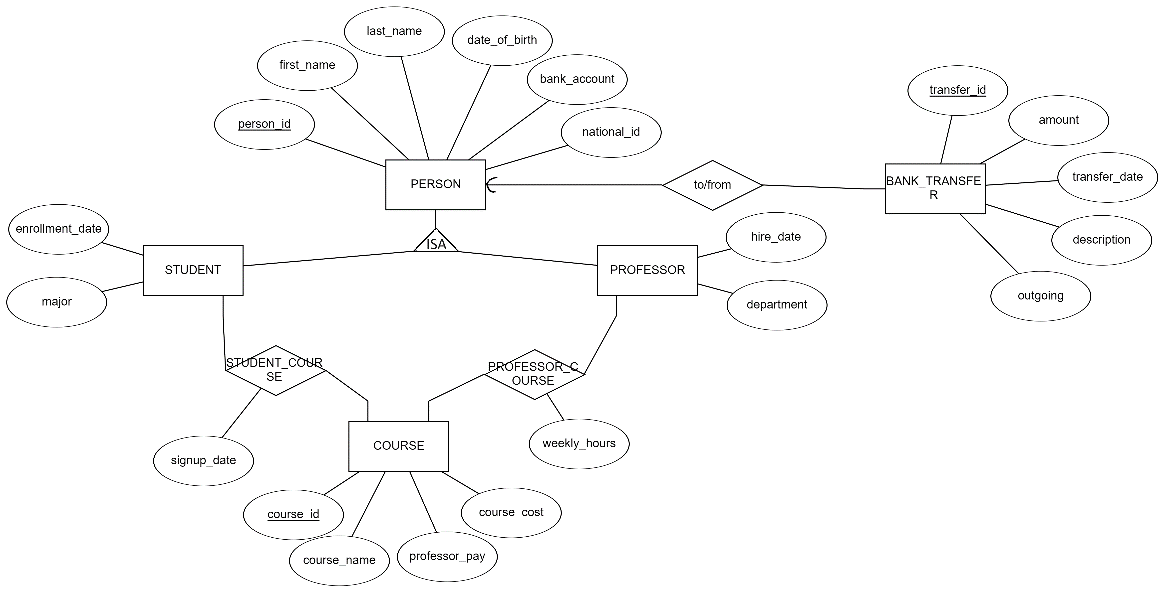
בעזרת בורא כל עולמים נתחיל שלב 4 בתפילה ובתחנונים

עיון קל במבנה הטבלאות יספק לנו את תרשים הDSD הבא:

תמונה שמכילה טקסט, תרשים, קו, מקביל

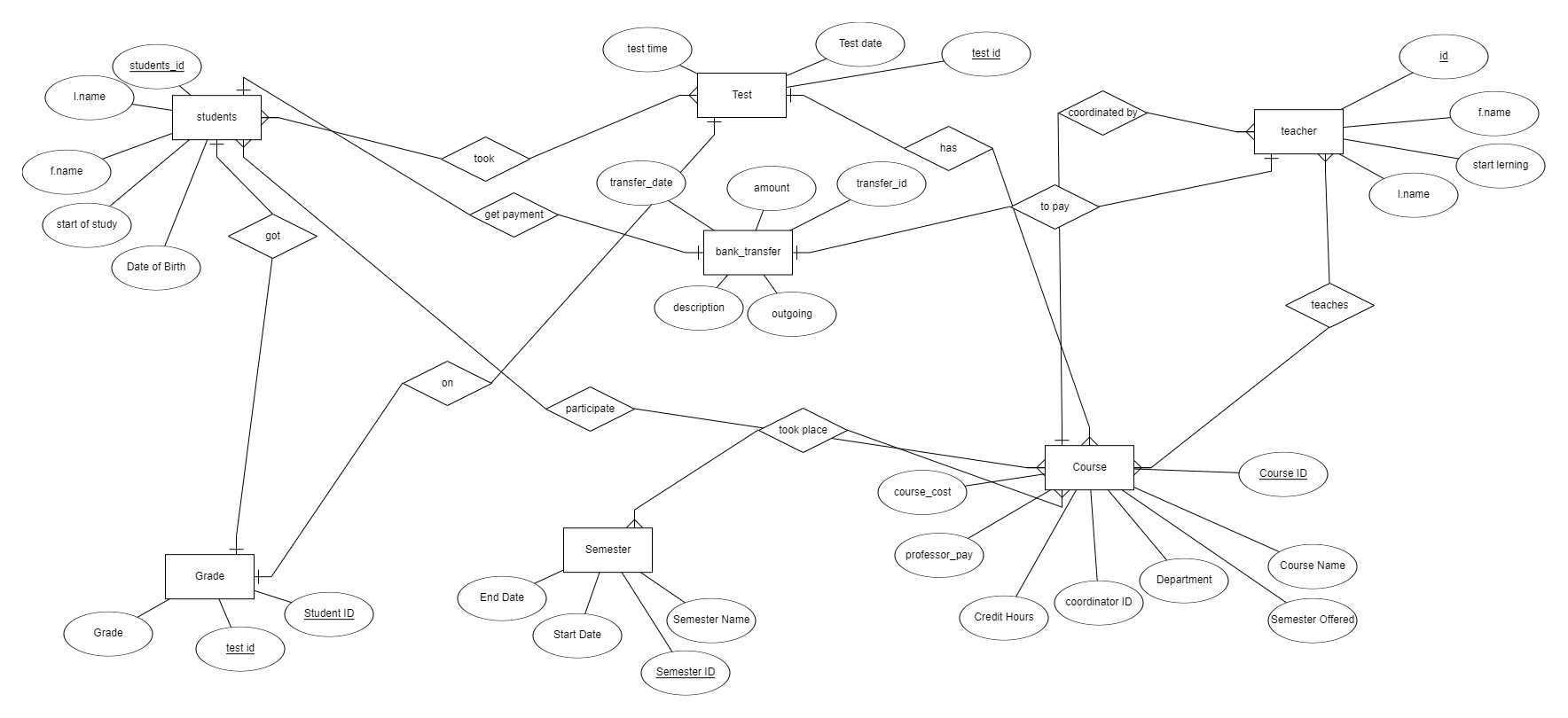
התיאור נוצר באופן אוטומטי

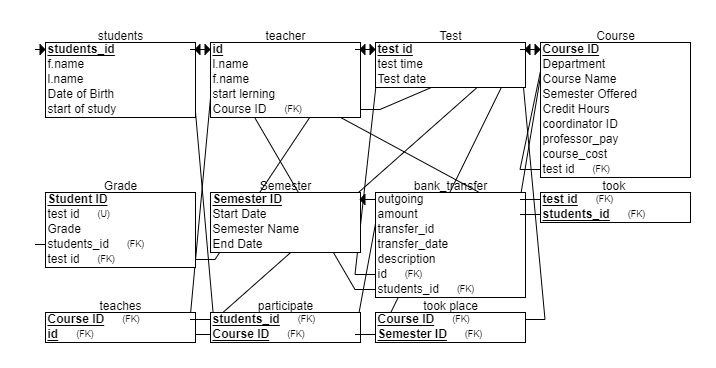
ובכמה הקלקות על העכבר קיבלנו את תרשים הERD:



כעת שלב ההחלטות. יש כמה יישויות שמשותפות לנו ולהם, ויש ביניהן כמה הבדלים דקים. ההבדל הגדול ביותר הוא שאצלם "סטודנט" ו"מרצה" יורשים מ"איש", ואצלנו לא. וכאן נתקלנו בבעיה, דממ"נ אם נעשה גם אנחנו יישות "איש", תהיה לנו בעיה עם הת"ז של הסטודנטים והמרצים, שאצלנו יתכן שלסטודנט ולמרצה יש אותו ת"ז, ועכשיו נצטרך לשנות את הנתונים שלנו עצמם, לא רק את הטבלאות. ואם לא נעשה ישות כזו, תהיה בעיה לייבא את הנתונים שלהם, כי ישות "העברות בנקאיות" שלהם מקושרת לישות "איש", ואיך נייבא את הנתונים שלהם? והחלטנו לא ליצור יישות "איש", אלא ליצור קשר העברה לסטודנט ועוד קשר בין העברה למרצה.

ואלו תרשימי הERD והDSD לאחר אינטגרציה:





לאחר שטיפלנו בבעיה העיצובית, נעבור לבעיה הטכנית העיקרית באינטגרציה: מה עושים כשיש טבלאות זהות בשני מסדי הנתונים? כשנאחד את הנתונים עלולה להיות התנגשות במפתחות הראשיים. נוכל לשנות את המפתחות הבעיתיים, אך זה יצור בעיה כאשר המפתח הזה הוא מפתח זר בטבלה אחרת. ובכן מה נעשה? נוסיף לכל מפתח זר את האילוץ ON UPDATE CASCADE (במקרה שלנו הכי קל לעשות זאת בקובץ הגיבוי שקיבלנו בעצמו. זה ודאי יותר קל מלשנות אח"כ בDB), אח"כ נשנה את כל המפתחות הראשיים לטווח שלא נמצא אצלנו (במקרה שלנו נכפיל ב10,000), ואז נמזג את הטבלאות.

בשביל לטפל בהבדל שאצלנו אין ירושה ובDB שקיבלנו יש, מיזגנו את הנתונים שלהם לתוך טבלה משותפת למוריש וליורש (ע"י שאילתא) עבור כל יורש, שתתאים לסכמה שלנו. ולעומת זאת טבלה שלהם שהיתה מקושרת רק למוריש, קיבלה אצלנו שתי טבלאות כדי להתקשר ליורשים.

שאר הדברים היו רק התאמות קטנות בין השדות שלנו לשלהם. ופרטיהם הלא הם כתובים בקובץ הקוד המצורף.