NYU, Tandon School of Engineering

October' 2024

CS-GY 6083

Principles of Database Systems Section A, Fall 2024

Homework #3

Submitted by:

Khwaab Thareja | N15911999 | Kt3180

Guided By: Prof Phyllis Frankl

Use the University DB, unless otherwise noted. Write SQL queries (unless otherwise noted) for each of the following:

1. Find IDs and names of students who got a B in CS-101 and an A in CS-319

Answer 1

```
Select distinct s.ID,name

from student s

inner join takes t on s.ID=t.ID

where t.course_id='CS-319' and Grade='A'

and s.id in

( Select s.ID from student s

inner join takes t on s.ID=T.ID

and t.course_id='CS-101' and Grade='B')
```

2. A. Create a table Gradepoint(grade,points) to associate letter grades with points and fill it with the appropriate values (’A’, 4.0), (’A-’ ,3.7), etc.

B. Add a foreign key constraint to another table, referencing thie Gradepoint table. Note which table, write the constraint, including “on delete” and “on update” clauses and briefly justify your choices for those clauses.

C. Define a VIEW GradePointAvg(ID, GPA) that lists each student’s ID and gradepoint average.

Note: If all courses had the same number of credits, you could compute gradepoint averages with a query involving a natural join of takes and gradepoint, along with the AVG aggregation operator, grouping by ids.However, that solution doesn’t take account of different courses having different numbers of credits. Assume that all graded courses are included, even if a student repeats the same course.

Answer 2.A

```
Drop table if exists GradePoint;

create table GradePoint

        (grade varchar(2),

         point float(4),

         primary key (grade)

        );

insert into GradePoint values ('A', 4.0);

insert into GradePoint values ('A-', 3.7);

insert into GradePoint values ('B+', 3.3);

insert into GradePoint values ('B', 3.0);

insert into GradePoint values ('B-', 2.7);

insert into GradePoint values ('C+', 2.3);

insert into GradePoint values ('C', 2.0);

insert into GradePoint values ('C-',1.7);

insert into GradePoint values ('D+', 1.3);

insert into GradePoint values ('D', 1.0);

insert into GradePoint values ('F', 0);
```

Answer 2B

```
ALTER TABLE Takes
ADD CONSTRAINT fk_grade
FOREIGN KEY (grade) REFERENCES Gradepoint(grade)
ON DELETE SET NULL
--  When a grade in the Gradepoint table is deleted, it will set the grade to NULL in the takes table, maintaining referential integrity while avoiding orphan records.
ON UPDATE CASCADE;
-- If a grade in the Gradepoint table is updated, the corresponding records in takes will automatically update, ensuring consistent grade values across the database.;
```

Answer 2C

```
DROP VIEW IF EXISTS GradePointAvg;
create view GradePointAvg as
Select distinct t.id,round(sum(point*credits)/sum(credits),2) as GPA from takes t
INNER join gradepoint gp on gp.grade=t.grade
INNER join course c on c.course_id=t.course_id
where t.grade is not null
GROUP BY T.ID;
Select * from GradePointAvg;
```

3. Find the name, ID, and GPA of the Comp. Sci. student who has the highest GPA among all Comp. Sci. students. If several students are tied for the highest, your query should return them all. Do not sort students by GPA

Answer 3

```
Select ID,name,GPA from
(
Select distinct s.id,
S.name,
round(sum(point*credits)/sum(credits),2) as GPA ,
rank() over (Order by sum(point*credits)/sum(credits) DESC) as rnk
from takes t
left join gradepoint gp on gp.grade=t.grade
left join course c on c.course_id=t.course_id
left join student s on s.ID=t.ID
where t.grade is not null and s.dept_name='Comp. Sci.'
GROUP BY T.ID,s.name
) a
where rnk=1;
```

4. Find the ID and name of each Comp. Sci. student who has not taken any courses offered by the Math department.

Answer 4

```
Select distinct s.ID,s.name from Student s

inner join takes t on t.ID=s.ID

inner join course c on t.course_id=c.course_id

where s.dept_name='Comp. Sci.' and c.dept_name not in ('Math') ;
```

;

5. Find ID, name, course_id of each student and each course they took in Fall 2009. Students who did not take any courses that semester should be listed with NULL as the course_id.

Answer 5

```
Select s.ID,name,t.course_ID from

student s

left join

(Select t.ID,t.course_id from takes t where semester='Fall' and year='2009') t on
t.ID=s.ID;
```

6. Find the ID, name, and Fall 2009 GPA for each student. Students who didn’t take any courses in Fall 2009 should be listed with GPA either NULL or zero.

Answer 6

```
Select s.ID, name, Fall_2009_GPA from student s

left join (

Select t.id,

round(sum(point*credits)/sum(credits),2) as Fall_2009_GPA

from takes t

inner join gradepoint gp on gp.grade=t.grade

inner join course c on c.course_id=t.course_id

Where t.grade is not null and t.year=2009 and t.semester='Fall'

GROUP BY T.ID) gpa on s.id=gpa.id;
```

7. Find the ID and name of each student who has taken every course taught by the instructor whose ID is 10101.

a. Write an SQL query using checks for empty set differences

b. Write an SQL query using comparison of sizes of sets

c. Write a TRC query. Hint: it should involve universal quantification.

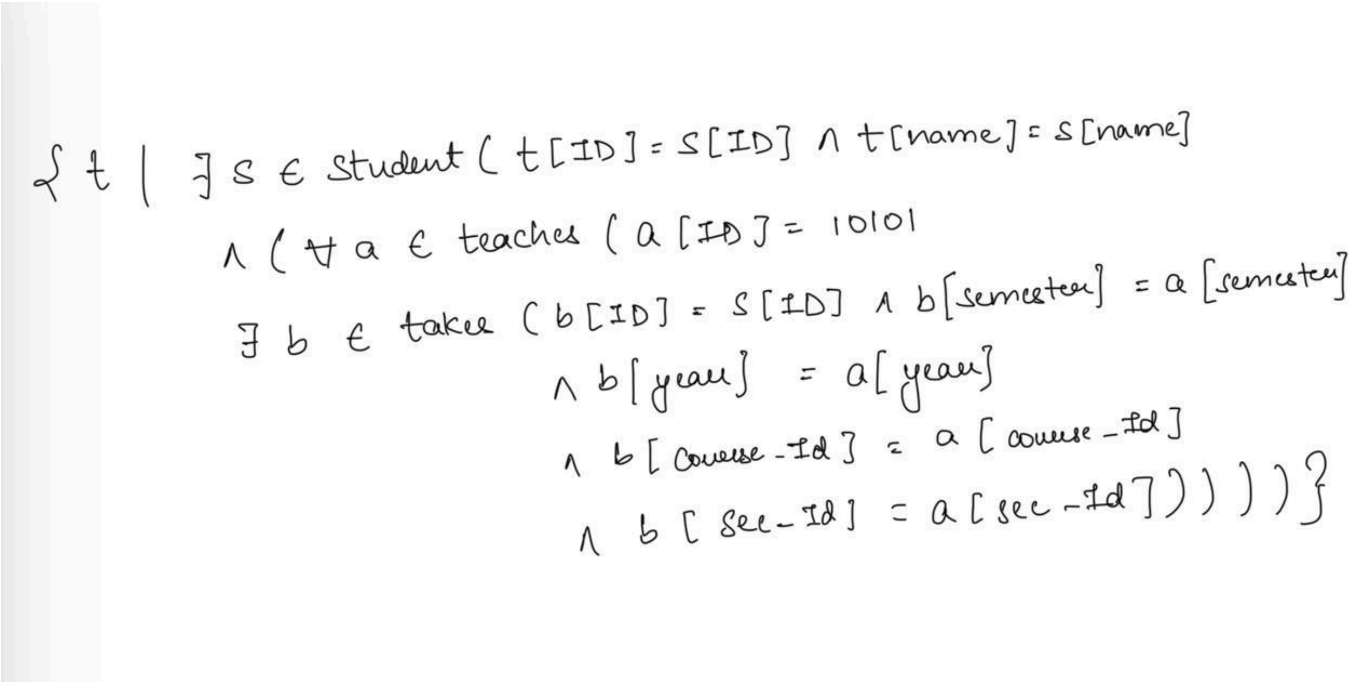
Answer 7 a

```
SELECT s.id, s.name
FROM student s
WHERE NOT EXISTS (
SELECT course_id
FROM teaches
WHERE ID = 10101
AND (course_id, sec_id, semester, year) NOT IN (
SELECT course_id, sec_id, semester, year
FROM takes
WHERE ID = s.ID
        )
);
```

Answer 7 b

```
          SELECT s.id, s.name
FROM student s
WHERE (
  SELECT COUNT(*)
  FROM teaches
  WHERE ID = 10101
) = (
  SELECT COUNT(*)
  FROM takes t2
  JOIN teaches t3 ON t2.course_id = t3.course_id
          AND t2.sec_id = t3.sec_id
          AND t2.semester = t3.semester
          AND t2.year = t3.year
  WHERE t2.ID = s.ID AND t3.ID = 10101
)
```

Answer 7 c



$$
\begin{aligned}
\{ t \mid\ & \exists s \in Student\ (t[ID] = s[ID] \wedge t[name] = s[name] \\
& \wedge (\forall a \in teaches\ (a[ID] = 10101 \\
& \exists b \in takes\ (b[ID] = s[ID] \wedge b[semester] = a[semester] \\
& \qquad \wedge b[year] = a[year] \\
& \qquad \wedge b[course\_Id] = a[course\_Id] \\
& \qquad \wedge b[sec\_Id] = a[sec\_Id])))) \}
\end{aligned}
$$

8. Using Retailer DB: Let’s define the “profitability” of a product to be the sum of the difference between priceEach and basePrice for all items ordered (taking quantity into account) divided by the total number of items of that product ordered. Write an SQL query to find the product name and product code of the product that has the highest profitability. (If there are ties, all such products should be listed. Do not use sorting.) Your solution should include WITH, VIEW, or TEMPORARY TABLE creations with comments to make them readable.

Answer 8

```
        Drop view if exists product_profit;

        Create view product_profit as

        select

        productname,

        productCode,

        profitability from

-- have to put in subquery because of rank function

        ( select

        productname,

        productCode,

 -- Calculating profitability : total_order_profit is derived from subquery

        sum(total_order_profit)/sum(quantityOrdered) as profitability,

        rank() over ( ORDER BY sum(total_order_profit)/sum(quantityOrdered) DESC)rnk

 -- Used rank function to rank based on profitability

        from

-- subquery to get profit per order for item which we could add later and divide by total quantity as did above

        (Select p.productname,
        od.productCode,
        (od.priceEach-p.buyPrice)*quantityOrdered as total_order_profit,
        quantityOrdered
        from orderdetails od
        left join products p on p.productCode=od.productCode
        ) as summary
        group by productCode) a where rnk =1
        ;

        Select * from product_profit;
```

9. Consider the posted solution to the tennis tournament problem from HW 1. Derive schemas and CREATE TABLE definitions for relevant tables and write constraints expressing the following:

a. A player cannot play a match against themself

b. Nomore than one match can be played on a given court during a given time slot

c. A player cannot play in more than one match during the same time slot

Try to determine whether your DBMS allows you to write those constraints in SQL and, if so, whether they are enforced by the DBMS. Briefly explain how you investigated this and your findings.

Answer 9.a

```
Drop table if exists Players;

CREATE TABLE Players (

   playerID INT PRIMARY KEY,

   name VARCHAR(100),

   age INT,

   ranking INT,

   seed INT,

   UNIQUE(seed)

);

Drop table if exists Events;

CREATE TABLE Events (

   eventID INT PRIMARY KEY,

   description VARCHAR(255),

   category ENUM('Mens Singles', 'Womens Singles', 'Juniors', 'Mixed Singles'),

   date DATE
```

```
);
Drop table if exists Courts;
CREATE TABLE Courts (
  courtNumber INT PRIMARY KEY,
  surface VARCHAR(50),
  location VARCHAR(100)
);
Drop table if exists TimeSlots;
CREATE TABLE TimeSlots (
  timeSlotID INT PRIMARY KEY,
  date DATE,
  startTime TIME,
  endTime TIME,
  UNIQUE(date, startTime) -- Ensures uniqueness of each time slot
);
Drop table if exists SinglesMatches;
CREATE TABLE SinglesMatches (
  matchID INT PRIMARY KEY,
  player1ID INT,
  player2ID INT,
  eventID INT,
  courtNumber INT,
  timeSlotID INT,
  score VARCHAR(20),
  highlights TEXT,

  FOREIGN KEY (player1ID) REFERENCES Players(playerID),

  FOREIGN KEY (player2ID) REFERENCES Players(playerID),

  FOREIGN KEY (eventID) REFERENCES Events(eventID),

  FOREIGN KEY (courtNumber) REFERENCES Courts(courtNumber),

  FOREIGN KEY (timeSlotID) REFERENCES TimeSlots(timeSlotID),

  CONSTRAINT chk_no_self_match CHECK (player1ID <> player2ID)  -- a) Prevents a player from playing against themselves

);
```

Answer 9.b

```
ALTER TABLE SinglesMatches

ADD CONSTRAINT unique_court_time UNIQUE (courtNumber, timeSlotID);
```

Answer 9.c

```
ALTER TABLE SinglesMatches

ADD CONSTRAINT unique_player1_time UNIQUE (player1ID, timeSlotID),

ADD CONSTRAINT unique_player2_time UNIQUE (player2ID, timeSlotID);
```