İstanbul Ün-Cerrahpaşa, Bilgisayar Müh. Böl., Veritabanı Yönetim Sistemleri, Güz 2021, Atakan Kurt

**ÖDEV 2: SQL,** Teslim tarih ve şekli: 02.12.2021 24:00, AKSİS

**KURALLAR:**

1. Ödevde 10 soru olup, her soru 1 puandır. Bu ödevin dönem notuna etkisi %10 dur. Yani alınan her puan dönem notuna 1 puan olarak yansır.
2. Ödevde kopya çekmek disiplin suçudur. Ödev de bir sorunun dahi kopya olması durumunda ödevin tamamı (kopya çeken ve kopya veren ayırımı yapılmaksızın) kopya sayılacak olup, not olarak -10 puan verilecektir. Yani dönem sonu toplam notunuzdan 10 puan silinecektir. Disipline verilmeniz durumunda desten doğrudan kalacaksınız.
3. Ödevler AKSIS üzerinden teslim edilecektir.
4. Geç ödev teslimi yapılmayacaktır. Ödevden haberi olmamak mazeret sayılmayacaktır.
5. Ödev PostgreSQL veritabanında yapılacaktır.
6. Cevapları gereksiz yere karmaşık yazmayınız. İhtiyaç duyulmayan tabloları sorguda gereksiz yere kullanmayınız. Bazı soruları sorguda kullanılması istenen operatörlerle (örnek: EXISTS, IN, =SOME vb.) ifade ediniz. Aykırı durumlarda not kırılacaktır.
7. Geçici ve sanal tablolara ve alan adlarına anlamlı isimler veriniz.
8. Her sorguyu ve sorgunun sonucunu (döndürülen kayıtları) kopyala-yapıştır ile veya başka bir şekilde bir metin veya MS Word dosyasına aktarınız. Bu dosyayı PDF olarak saklayabilirsiniz.
    a. Ödev çözümünün başında dersin kodu, adı, öğrenci numaranız, adınız ve ödev numarası (1. ödev), hangi veritabanı sistemi ile yapıldığı yazılı olmalıdır.
    b. Her sorgu için soru numarası, sorgunun kendisi ve sorgu sonucu ödeve konulmalıdır. Sorgunuzu ve alt sorgularınızı her bir SELECT/FROM/WHERE/GROUP BY/HAVING/ORDER BY/… cümleciği farklı satırlarda alt alta gelecek şekilde düzgün olarak yazınız veya biçimlendiriniz (indentation). Anahtar kelimeler büyük harfle, tablo ve alan adları küçük harfle yazılı olmalıdır.

**VERİTABANI ŞEMASI**

Student (sid, name, did, noOfCourses, GPA)// ogrenci(ogrenci-no, adi, bolum-no, dersSayisi)
Take (sid, cid, grade) // ders-al(ogrenci-no, ders-kodu, notu)
Course (cid, title, credits, did, noOfStudents)// ders(ders-kodu, adi, kredisi, bolum-no, ogrSayisi)
Department (did, name, noOfStudents) // bolum(bolum-no, adi, ogrSayisi)
Teacher (tid, name, placeOfBirth, did) // hoca(hoca-no, adi, dogum-yeri, bolum-no)
Teach (tid, cid) // ders-ver(hoca-no, ders-kodu)

**SORGULAR**

(Önce yukarıdaki tabloları CREATE TABLE ile oluşturunuz. Sonra tablolara rastgele kayıtlar INSERT ediniz. Sonra sorgularınızı yazıp test ediniz.

1. Tüm öğrencilerin ağırlıklı not ortalamalarını (GPA) bir alt sorguyla hesaplayıp güncelleyiniz.Açıklama: Şu komuttaki alt sorguyu yazmanız gerekecektir. UPDATE student SET GPA = (…..)

   GPA = SUM(grade x credits)/SUM(credits) formülüyle hesaplanır.

```
UPDATE student SET GPA = (SELECT SUM(grade*credits)/SUM(credits)
                          FROM take t, course c
                          WHERE t.cid=c.cid AND t.sid=student.sid)
```

Dikkat bu sorguda WHERE cümlesi tüm kayılar güncelleneceği için kullanılmamıştır. Alt sorguda dış sorgudaki student tablosu correlated bir sorgu yazılmıştır.

2. Bölümlerin hepsinde ders veren hocaların kayıtlarını listeyeyiniz. Açıklama: Bu bir ilişkisel cebir bölme işlemidir. Bu soru şu sorgu kalıbıyla cevaplanabilir:

SELECT * FROM teacher t WHERE ((tüm dersler) – (t hocasının verdiği dersler) = ϕ)

```
SELECT *
FROM teacher t
WHERE NOT EXISTS((SELECT cid FROM course) MINUS (SELECT cid FROM teach x WHERE x.cid=t.cid))
```

3. Hocaların verdiği ders sayılarının ortalamasının altında ders veren hocaların "tid, name, verdiği ders sayısı ve derslerini alan öğrencilerin sayılarını" listeleyiniz. Açıklama: Bu soru 3 adımda çözülebilir.

   (i) Her hocanın kaç ders verdiği bulunur: dersSayi(tid, dersSayisi)
   (ii) dersSayi sorgusundaki dersSayisi' değerlerinin AVG ile ortalaması bulunur: ortalamaDers(ort)
   (iii) dersSayi, ortalamaDers ve teacher tablolarından dersSayisi > ortalamaDers.ort olan kayıtlar için soruda istenenler hesaplanır

```
WITH
dersSayi(tid, dersSayisi) AS
        SELECT tid, COUNT(cid)
        FROM teach
        GROUP BY tid
ortalamaDers(ort) AS
        SELECT AVG(dersSayisi)
        FROM dersSayi,
SELECT t.tid, t.name, dersSayisi, (SELECT COUNT(sid)
                                  FROM take y, teach z
                                  WHERE y.cid=z.cid AND z.tid=t.tid)
FROM teacher t, dersSayi d, ortalamaDers o
WHERE t.tid=d.tid AND d.dersSayisi < o.ort;
```

Yukarıdaki çözümde bir hocanın "derslerini alan öğrencilerin sayısı" SELECT içindeki bir alt sorguyla bulunmuştur. Fakat bu aşağıdaki çözümdeki gibi 1. Adımda da hesaplanabilirdi:

```
WITH
dersSayi(tid, dersSayisi, orgrenciSayisi) AS
        SELECT tid, COUNT(DISTINCT t1.cid), COUNT(sid) -- COUNT(DISTINCT t2-.cid) de yazılabilirdi.
        FROM teach t1, take t2
        WHERE t1.cid=t2.cid
        GROUP BY t1.tid
ortalamaDers(ort) AS
        SELECT AVG(dersSayisi)
        FROM dersSayi,
SELECT t.tid, t.name, dersSayisi, orgrenciSayisi
FROM teacher t, dersSayi d, ortalamaDers o
WHERE t.tid=d.tid AND d.dersSayisi < o.ort;
```

Yukarıda DISTINCT zorunludur, bir dersi alan birçok öğrenci olduğundan dolayı(?)

4. Kredisi > 3 olan ve en fazla 20 öğrencinin aldığı derslerin "cid, ogrenci sayısı, not ortalama"larını dersi alan öğrenci sayısına göre artan, dersteki notların ortalamasına göre azalan sırada listeleyiniz. Açıklama: Bu sorguda SELECT FROM WHERE GROUP BY HAVING ORDER BY cümleciklerinin hepsini kullanmanız gerekmektedir.

```
SELECT c.cid, COUNT(sid) sayi, AVG(grade) ort
FROM course c, take t
WHERE c.cid=t.cid AND credits > 3
GROUP BY c.cid
HAVING COUNT(sid) < 21
ORDER BY sayi ASC, ort DESC
```

5. Aldığı tüm derslerdeki notu o dersteki (kendi notu hariç olmak üzere) notların ortalamasından yüksek olan öğrencilerin sid'lerini listeleyiniz.

Soruyu "aranan kümenin dışındaki (yani en az bir dersinde notu ders ortalamasının altındaki) öğrencileri bulup", bu kümeyi "ders alan öğrencilerin kümesinden" **çıkararak** tersinden çözebiliriz:

```
WITH adim1(sid) AS --en az bir dersinde notu ders ortalamasının altındaki öğrencilerin kümesi
        SELECT sid
        FROM take t
        WHERE grade <= (SELECT AVG(grade) -- t.sid'nin notu hariç dersteki notların ortalaması
                        FROM take x
                        WHERE x.cid=t.cid AND x.sid!=t.sid)
(SELECT sid FROM take) MINUS (SELECT sid FROM adim1);
```

6. Bölümlerin "did, öğrenci sayılarını ve bölümdeki öğrencilerin notlarının ortalamalarını" listeleyiniz. Bu soruda öğrenci sayısı için SELECT cümleciğinde alt-sorgu kullanılacaktır. Açıklama: SELECT did, (alt sorgu) "ogrenci sayisi", (alt sorgu) ogrenciNotOrtalama FROM department d. gibi yazılır.

```
SELECT did,
        (SELECT COUNT(sid) FROM department d1 WHERE d1.did=d.did) "ogrenci sayisi",
        (SELECT AVG(grade) FROM student s, take t WHERE s.did=d.did AND s.sid=t.sid) ogrNotOrt
FROM department d
```

7. En az iki farklı hocadan yada iki farklı bölümden ders alan öğrencilerin kayıtlarını listeleyiniz. Açıklama: "yada" dendiği için UNION kullanıp 2 alt sorgu yazılabilir. Alt sorgularda COUNT(DISTINCT teach.tid) ve COUNT(DISTINCT course.did) cümlecikleriyle sayma işlemi yapacaktır. İki farklı hoca için teach ve take tablolarına erişim, iki farklı ders için take ve course tablolarına erişim gereklidir.

```
WITH
ikiFarkliHoca AS
        (SELECT sid
        FROM take t, teach h
        WHERE t.cid=h.cid
        GROUP BY t.sid
        HAVING COUNT(DISTINCT h.tid) > 1)
ikiFarkliBolum AS
        (SELECT sid
        FROM take t, course c
        WHERE t.cid=c.cid
        GROUP BY t.sid
        HAVING COUNT(DISTINCT c.did) > 1)
SELECT s.*
FROM student s, ((SELECT * FROM ikiFarkliHoca) UNION (SELECT * FROM ikiFarkliBolum)) x
WHERE s.sid=x.sid
```

8. "Ali KURT" ve "Ayse KURT" adlı öğrencilerin aldığı derslerin kredilerinin toplamında daha fazla kredi alan öğrencilerin kayıtlarını listeleyiniz. Açıklama: Birinci adımda öğrenciye göre gruplama yapıp kredi toplamı diğer 2 öğrencinin kredi toplamından büyük olan grupların sid'leri listelenir. İkinci adımda student tablosundan bu öğrencilerin kayıtları listelenir.

```
WITH krediToplam AS
        SELECT SUM(credits) toplam
        FROM student s, take t, course c
        WHERE (s.name='Ali KURT' OR s.name='Ayse KURT') AND s.sid=t.sid AND t.cid=c.cid
SELECT s.*
FROM student s, take t, course c
```

```
WHERE s.sid=t.sid AND t.cid=c.cid
GROUP BY s.sid
HAVING SUM(credits) > (SELECT toplam
                       FROM krediToplam)
```

9. Aynı notu 2 farklı dersten almamış yani tüm notları birbirinden farklı olan öğrencilerin kayıtlarını ~~WHERE içinde UNIQUE fonksiyonu kullanarak~~ veriniz. PostgreSQL UNIQUE fonksiyonunun desteklemediği için soru değiştirildi.

"Ders alan öğrenciler kümesinden" (yada tüm öğrenciler kümesinden) "bir notu 2 farklı dersten almış öğrenciler kümesi" çıkarılır.

```
WITH 1not2ders AS
        SELECT sid
        FROM take t
        GROUP BY sid, grade
        HAVING COUNT(cid)  > 1
SELECT s.*
FROM student s, ((SELECT sid FROM take) MINUS (SELECT sid FROM 1not2ders)) x
WHERE s.sid=x.sid
```

10. Girilmemiş yada verilmemiş notu olmayan (IS NOT NULL) öğrencilerin kayıtlarını listeleyiniz

Ders alan öğrenci kümesinden (yada tüm öğrencilerin kümesinden) en az bir notu NULL olan öğrencilerin kümesi çıkarılır. Yada NOT EXISTS kullanılarak NULL notu olmayan öğrenciler listelenerek bulunabilir. 2. çözüm verilmiştir.

```
SELECT *
FROM student s
WHERE NOT EXISTS    (SELECT *
                     FROM take t
                     WHERE t.sid=s.sid AND grade IS NULL)
```