

BIMU3064-Veri Tabanı Yönetim Sistemleri

**İBRAHİM BAŞAR YARGICI**

**1306191467**

Ödev2

Veritabanı = PostgreSQL

(02.12.2021)

**Veri Tabanı ve Veri Tabanı Şeması Hazırlığı**

|  |
| --- |
| CREATE DATABASE "Homework\_0s2122021";  DROP TABLE IF EXISTS Department CASCADE;  DROP TABLE IF EXISTS Student CASCADE;  DROP TABLE IF EXISTS Take CASCADE;  DROP TABLE IF EXISTS Course CASCADE;  DROP TABLE IF EXISTS Teacher CASCADE;  DROP TABLE IF EXISTS Teach CASCADE;  CREATE TABLE Department(  did INTEGER,  name VARCHAR(50),  noOfStudents INTEGER,  PRIMARY KEY(did)  );  CREATE TABLE Student(  sid INTEGER,  name VARCHAR(50),  did INTEGER NOT NULL,  noOfCourses INTEGER,  GPA FLOAT(2),  PRIMARY KEY(sid),  FOREIGN KEY(did) REFERENCES Department(did)  );  CREATE TABLE Course(  cid INTEGER,  title VARCHAR(50),  credits INTEGER,  did INTEGER NOT NULL,  noOfStudents INTEGER,  PRIMARY KEY(cid),  FOREIGN KEY(did) REFERENCES Department(did)  );  CREATE TABLE Take(  sid INTEGER,  cid INTEGER,  grade INTEGER,  PRIMARY KEY(sid,cid),  FOREIGN KEY(sid) REFERENCES Student(sid),  FOREIGN KEY(cid) REFERENCES Course(cid)  );  CREATE TABLE Teacher(  tid INTEGER,  name VARCHAR(50),  placeOfBirth VARCHAR(50),  did INTEGER NOT NULL,  PRIMARY KEY(tid),  FOREIGN KEY(did) REFERENCES Department(did)  );  CREATE TABLE Teach(  tid INTEGER,  cid INTEGER,  PRIMARY KEY(tid,cid),  FOREIGN KEY(tid) REFERENCES Teacher(tid),  FOREIGN KEY(cid) REFERENCES Course(cid)  );  INSERT INTO Department VALUES  (1,'Bil Müh.',5),  (2,'Elektrik Elektronik Müh.',3);  INSERT INTO Course VALUES  (1,'Programlama Giriş 1',4 ,1,2),  (2,'Programlama Giriş 2',6 ,1,2),  (3,'Programlama Giriş 3',8 ,1,1),  (4,'Programlama Giriş 4',12,1,0),  (5,'Devreler 1',4,2,2),  (6,'Devreler 2',6,2,1),  (7,'Programlama Giriş 5',16,1,0),  (8,'Devreler 3',8,2,0);  INSERT INTO Student VALUES  (1,'basar',1,2,4.00),  (2,'elif',1,1,2.12),  (3,'ahmet',1,1,3.05),  (4,'beyazıd',1,1,3.85),  (5,'ali',1,0,1.91),  (6,'veli',2,2,2.45),  (7,'ibrahim',2,1,3.49),  (8,'aysun',2,0,1.2);  INSERT INTO Take VALUES  (1,1,90),  (1,2,95),  (2,1,24),  (3,2,24),  (4,3,24),  (6,5,90),  (6,6,95),  (7,5,95);  INSERT INTO Teacher VALUES  (1,'atakan','ankara',1),  (2,'mehmet','istanbul',1),  (3,'zeynep','batman',1),  (4,'idil','izmir',2),  (5,'orkun','kocaeli',2),  (6,'Haydar','Keke',1);  INSERT INTO Teach VALUES  (1,1),  (1,4),  (2,3),  (3,2),  (4,6),  (5,5),  (6,7),  (6,8);  SELECT \* FROM course;  SELECT \* FROM department;  SELECT \* FROM student;  SELECT \* FROM teacher;  SELECT \* FROM teach;  SELECT \* FROM take; |

**Sorgulara Giriş**

Sorgu 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 s  SET GPA = (SELECT SUM(take.grade \* course.credits) / SUM(course.credits)  FROM student s2  INNER JOIN take ON take.sid = s2.sid  INNER JOIN course ON course.cid = take.cid  WHERE s.sid = s2.sid  GROUP BY s2.sid) |

metin, iç mekan, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, televizyon, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

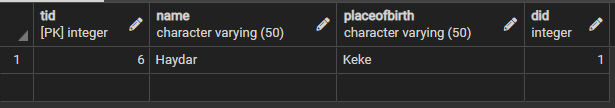
Note: sid = 5 and sid = 8’s gpa is null because of they are not taking any course!

Sorgu 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 ((tüm bölümler) – (t hocasının verdiği derslerin  -- bölümleri) = φ)  SELECT \* FROM teacher t  WHERE NOT EXISTS(  (  SELECT d.did  FROM Department d  )  EXCEPT  (  SELECT d.did  FROM Teach te, Course c, Department d  WHERE t.tid = te.tid AND te.cid = c.cid AND c.did = d.did  )  ) |

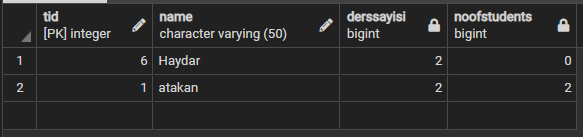


Sorgu 3:

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.

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

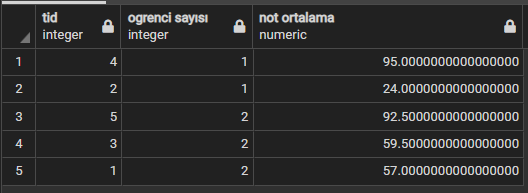
|  |
| --- |
| WITH ds AS  (SELECT teacher.tid as tid, COUNT(teacher.tid) as dersSayisi, teacher.name  FROM teacher INNER JOIN teach ON teacher.tid = teach.tid  GROUP BY teacher.tid),  od AS  (SELECT AVG(ds.dersSayisi) as ort  FROM ds),  cs AS  (SELECT SUM(course.noofstudents) as noofstudents,teach.tid  FROM teach, course  WHERE teach.cid = course.cid  GROUP BY teach.tid  ),  rs AS (  SELECT ds.tid, ds.name, ds.dersSayisi  FROM ds, od  WHERE ds.dersSayisi > od.ort  )  SELECT rs.tid, rs.name, rs.dersSayisi,cs.noofstudents  FROM rs, cs  WHERE rs.tid = cs.tid |



Sorgu 4:

Kredisi > 3 olan ve en fazla 20 öğrencinin aldığı derslerin "tid, 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.

|  |
| --- |
| WITH cbt3 AS  (SELECT c.cid, c.noofstudents  FROM course c  WHERE c.credits > 3),  t AS  (SELECT cbt3.cid,cbt3.noofstudents, te.tid  FROM teach as te,cbt3  WHERE te.cid = cbt3.cid),  temp AS  (SELECT AVG(take.grade),t.cid,t.noofstudents  FROM take,t  WHERE take.cid = t.cid  GROUP BY t.cid,t.noofstudents  HAVING t.noofstudents < 20  )  SELECT t.tid, temp.noofstudents as "ogrenci sayısı", temp.avg as "not ortalama"  FROM t,temp  WHERE temp.cid = t.cid  ORDER BY temp.noofstudents ASC, temp.avg DESC |



Sorgu 5

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

|  |
| --- |
| SELECT \*  FROM student s  WHERE EXISTS(  (  SELECT t.grade  FROM take t  WHERE s.sid = t.sid AND t.grade >=  (  SELECT AVG(t.grade)  FROM take t  WHERE s.sid = t.cid  )  )  ) |

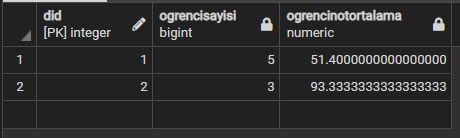
metin, ekran görüntüsü, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sorgu 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.

|  |
| --- |
| -- There are two different solutions:  WITH dcid AS  (SELECT co.cid, t.grade, d.did, t.sid  FROM course co INNER JOIN department d ON d.did = co.did  INNER JOIN take t ON co.cid = t.cid)  SELECT d.did,  (  SELECT COUNT(sid) as ogrenciSayisi  FROM dcid  GROUP BY dcid.did  HAVING d.did = did  ),  (  SELECT AVG(grade) as ogrenciNotOrtalama  FROM dcid  GROUP BY did  HAVING d.did = did  )  FROM department d  -- Or  WITH dcid AS  (SELECT co.cid, t.grade, d.did, t.sid  FROM course co INNER JOIN department d ON d.did = co.did  INNER JOIN take t ON co.cid = t.cid),  notavg AS  (SELECT AVG(grade) as ogrenciNotOrtalama, did  FROM dcid  GROUP BY did  ),  scount AS  (SELECT COUNT(sid) as ogrenciSayisi, did  FROM dcid  GROUP BY dcid.did  )  SELECT d.did, scount.ogrenciSayisi, notavg.ogrenciNotOrtalama  FROM department d, notavg, scount  WHERE d.did = notavg.did AND scount.did = d.did |



Sorgu 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 twoTeacher AS  (  SELECT te.sid  FROM teach t,take te  WHERE t.cid = te.cid  GROUP BY te.sid  HAVING COUNT(te.sid)>=2  ),  twoCourse AS  (  SELECT ta.sid  FROM take ta,course co  WHERE ta.cid = co.cid  GROUP BY ta.sid  HAVING COUNT(ta.cid)>=2  )  SELECT s.\*  FROM student s  WHERE s.sid IN(  (  SELECT t.sid  FROM twoTeacher t  )  UNION  (  SELECT t.sid  FROM twoCourse t  )  ) |

metin, ekran görüntüsü, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sorgu 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 sc AS  (  SELECT co.credits, t.sid  FROM course co INNER JOIN take t ON co.cid = t.cid  ),  ak AS  (  SELECT s.sid  FROM student s  WHERE s."name" = 'elif' OR s."name" = 'ibrahim'  -- soruda istenen isimin ali kurt ve ayse kurt olması ancak kendi verisetime  -- göre sorgunun düzgün çalışması için bu şekilde kullandım. Olması gerken:  -- WHERE s."name" = 'Ali KURT' OR s."name" = ‘Ayse KURT'  ),  akSum AS  (  SELECT SUM(sc.credits) as "sum"  FROM ak,sc  WHERE ak.sid = sc.sid  ),  allCredits AS  (  SELECT sc.sid, SUM(sc.credits) as "sum"  FROM student s, sc  WHERE s.sid = sc.sid  GROUP BY sc.sid  )  SELECT allcredits.sid  FROM akSum,allcredits  WHERE allcredits.sum > akSum.sum |

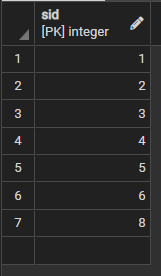
metin, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sorgu 9:

Aynı notu 2 farklı dersten almamış yani tüm notları birbirinden farklı olan öğrencilerin kayıtlarını WHERE içinde UNIQUE fonksiyonu kullanarak veriniz.

|  |
| --- |
| -- İki aynı notu alan yoktu. Bundan dolayı yeni kurs ve aynı not değerleri sid'si 7 olan öğrenciye atandı.  INSERT INTO take VALUES (7,6,95);  WITH hasSame AS  (  SELECT t.grade,t.sid  FROM take t  GROUP BY t.grade,t.sid  HAVING COUNT(t.grade)>1  )  SELECT s.sid  FROM student s, hasSame  WHERE hasSame.sid != s.sid |



Sorgu 10:

Girilmemiş yada verilmemiş notu olmayan (IS NOT NULL öğrencilerin listeleyiniz

|  |
| --- |
| -- Girilmemiş yada verilmemiş notu olan yoktu. Bundan dolayı girilmemiş notlu değer olması için sid'si 7 olan öğrenciye atandı.  INSERT INTO take VALUES (7,1,null);  SELECT t.grade, t.sid  FROM student s INNER JOIN take t ON s.sid = t.sid  WHERE t.grade IS NOT NULL |

