YILDIZ TEKNİK ÜNİVERSİTESİ

Orta Seviye SQL

Öğr. Gör. Dr. Yasemin Topuz

Yıldız Teknik Üniversitesi

BLM3041 – Veritabanı Yönetimi

24 Ekim 2025 | YTÜ Bilgisayar Mühendisliği

Neler konuşacağız?

- SQL (Structured Query Language)
- Aggregate Fonksiyonları
- İç içe (Nested) Alt Sorgular
- SET Operasyonları
- Verinin Oluşturulması ve Değiştirilmesi
  - INSERT
  - UPDATE
  - DELETE

SQL – Aggregate Fonksiyonları

- Bu işlevler, bir ilişkinin bir sütunundaki değerlerin çoklu kümesi üzerinde çalışır ve bir değer döndürür.

- **AVG**: ortalama değer
- **MIN**: minimum değer
- **MAX**: maksimum değer
- **SUM**: değerlerin toplamı
- **COUNT**: değer sayısı

SQL – Aggregate Fonksiyonları

- Kadın çalışanların ortalama maaşını bulunuz.

```
SELECT AVG(Salary) from EMPLOYEE WHERE sex='F';
```

	avg numeric
1	31000.000000000000

- 4 numaralı departman tarafından yürütülen projelerin kaç farklı lokasyonda olduğunu bulunuz.

```
SELECT COUNT(plocation) FROM PROJECT WHERE dnum=4;
SELECT COUNT(DISTINCT plocation) from PROJECT WHERE dnum=4;
```

	pname character varying (1	pnumber [PK] intege	plocation character vary	dnum integer
1	ProductX	1	Bellaire	5
2	ProductY	2	Sugarland	5
3	ProductZ	3	Houston	5
4	Computerization	10	Stafford	4
5	Reorganization	20	Houston	1
6	Newbenefits	30	Stafford	4

SQL – Aggregate Fonksiyonları

	essn [PK] character (9)	pno [PK] inte	hours numeric (3,1
1	333445555	20	10.0
2	987654321	20	15.0
3	888665555	20	[null]

- 20 numaralı projede toplam kaç saat çalışıldığını bulunuz.

```
SELECT SUM(hours) from WORKS_ON WHERE pno=20;
```

	sum numeric
1	25.0

- 20 numaralı projede çalışanların sayısını bulunuz.

```
SELECT COUNT(*) AS "Çalışan Sayısı" FROM WORKS_on WHERE pno=20
```

	Çalışan Sayısı bigint
1	3

```
SELECT COUNT(hours) FROM WORKS_on WHERE pno=20
```

	count bigint
1	2

SQL – Aggregate Fonksiyonları – Group By

- Her bir departmandaki çalışanların ortalama ve toplam maaşlarını bulunuz.

```
SELECT dno, AVG(Salary) AS "Ortalama Maaş", SUM (Salary) AS "Toplam Maaş"
FROM  employee GROUP BY dno
```

	dno integer	Ortalama Maaş numeric	Toplam Maaş numeric
1	5	33250.0000000	133000.00
2	4	31000.0000000	93000.00
3	1	55000.0000000	55000.00

- Aggregate fonksiyonları dışındaki öznitelikler, gruplama listesinde yer almalıdır.

```
/* hatalı sorgu */
SELECT dno, fname, AVG(Salary) AS "Ortalama Maaş" FROM EMPLOYEE GROUP BY dno;
```

SQL – Aggregate Fonksiyonları – Group By

- Her bir departmandaki çalışanların cinsiyete göre en düşük ve en yüksek maaşlarını bulunuz. Departman no ve cinsiyete göre A'dan Z'ye sıralayınız.

```
SELECT dno, sex, MIN(Salary) AS "En Düşük Maaş", MAX (Salary) AS "En Yüksek Maaş"
FROM  employee GROUP BY dno,sex ORDER BY dno, sex
```

	dno integer	sex charact	En Düşük Maaş numeric	En Yüksek Maaş numeric
1	1	M	55000.00	55000.00
2	4	F	25000.00	43000.00
3	4	M	25000.00	25000.00
4	5	F	25000.00	25000.00
5	5	M	30000.00	40000.00

SQL – Aggregate Fonksiyonları – Group By

- Her bir projede en az 20 saat çalışan personellerin sayısını ve toplam kaç saat çalıştıklarını listeleyiniz.

```
SELECT pno, COUNT(essn), SUM (Hours) FROM WORKS_ON
WHERE hours >= 20 GROUP BY pno;
```

	pno integer	count bigint	sum numeric
1	1	2	52.5
2	2	1	20.0
3	3	1	40.0
4	10	1	35.0
5	30	2	50.0

SQL – Aggregate Fonksiyonları – Group By

- Her bir personelin isimlerini ve görev aldıkları projelerin sayısını bulunuz.

```
SELECT e.ssn, e.Fname, COUNT(w.pno) FROM Employee e, Works_on w
WHERE e.ssn = w.essn GROUP BY e.ssn, e.Fname;
```

- Aynı sorgu;

```
SELECT e.ssn, e.Fname, COUNT(w.pno)
FROM Employee e JOIN Works_on w
ON e.ssn = w.essn GROUP BY e.ssn, e.Fname
```

- Bu sorgu projede görev almayan personelleri göstermez. Bunun yerine;

```
SELECT e.ssn, e.Fname, COUNT(w.pno)
FROM Employee e FULL JOIN Works_on w
ON e.ssn = w.essn GROUP BY e.ssn, e.Fname
```

Bu sorguda, 5 nolu kayıt dönmez.

Bu sorguda, 5 nolu kayıt ta döner.

	ssn [PK] character (	fname character v	count bigint
1	666884444	Ramesh	1
2	333445555	Franklin	4
3	999887777	Alicia	2
4	987987987	Ahmad	2
5	888665555	James	0
6	123456789	John	2
7	987654321	Jennifer	2
8	453453453	Joyce	2

SQL – Aggregate Fonksiyonları – Having

- En az 30.000 maaş alan birden fazla çalışanı olan yöneticileri listeleyiniz.

```
SELECT super_ssn, COUNT (ssn) as "Personel_Count" FROM employee
WHERE salary>=30000 and Personel_Count>1 GROUP BY super_ssn
```

```
SELECT super_ssn, COUNT (ssn) as "Personel_Count" FROM employee
WHERE salary>=30000 GROUP BY super_ssn HAVING COUNT(ssn)>1
```

	super_ssn character (9)	Personel_Count bigint
1	333445555	2
2	888665555	2

SQL – Aggregate Fonksiyonları – Having

- Toplam maaşı 75000'den büyük olan departmanları bulunuz.

```
SELECT dno, SUM(Salary) AS "Toplam Maaş"
FROM EMPLOYEE GROUP BY dno HAVING SUM(Salary)>75000;
```

	dno integer	Toplam Maaş numeric
1	5	133000.00
2	4	93000.00

- Not: Group by olmadan Having kullanılamaz. Having koşulları gruplar oluşturulduktan sonra uygulanırken, Where koşulları, gruplar oluşturulmadan önce uygulanır.

SQL – İç içe (Nested) Alt Sorgular

- SQL, alt sorguların iç içe kullanılmasına imkân sağlar. Alt sorgu (subquery), başka bir sorgunun içinde yer alan select–from–where ifadesidir.
- Aşağıdaki genel SQL kalıbında iç içe kullanım yapılabilir:

  SELECT A1, A2, ..., An

  FROM r1, r2, ..., rm

  WHERE P

- **FROM bölümü:** r_i , geçerli herhangi bir alt sorgu ile değiştirilebilir.
- **WHERE bölümü:** P, şu formda bir ifadeyle değiştirilebilir:

  B <işlem> (alt_sorgu)

  Burada B bir özniteliktir; <işlem> **IN** vb. ifadelerdir.
- **SELECT bölümü:** A_i, tek bir değer üreten bir alt sorgu ile değiştirilebilir.

SQL – İç içe (Nested) Alt Sorgular

- En az 1 çalışanın olduğu departmanları bulunuz.

```
SELECT dname, dnumber from DEPARTMENT
WHERE dnumber IN (SELECT  dno from EMPLOYEE)
```

```
SELECT d.dname,d.dnumber from DEPARTMENT d
WHERE EXISTS (SELECT 1 from EMPLOYEE e WHERE d.dnumber=e.dno)
```

$r \Leftrightarrow r \neq \emptyset$

- Hiç projesi olmayan departmanları bulunuz.

```
SELECT d.dnumber, d.dname FROM department d
WHERE d.dnumber NOT IN
(SELECT p.dnum FROM project p);
```

```
SELECT d.dnumber, d.dname FROM department d
WHERE NOT EXISTS
(SELECT 1 FROM project p WHERE p.dnum = d.dnumber);
```

$r \Leftrightarrow r = \emptyset$

SQL – SET (Küme) Operasyonları – Dahil Olma

- Hem projesi hem de çalışanı olan departmanları bulunuz.

```
SELECT d.dnumber FROM department d
WHERE d.dnumber IN (SELECT e.dno  FROM employee e)
 AND d.dnumber IN (SELECT p.dnum FROM project p);
```

```
SELECT e.dno from employee e
INTERSECT
SELECT p.dnum from project p
```

- Çalışanı olup projesi olmayan departmanları bulunuz.

```
SELECT d.dnumber FROM department d
WHERE d.dnumber IN (SELECT e.dno  FROM employee e)
  AND d.dnumber NOT IN (SELECT p.dnum FROM project p);
```

```
SELECT e.dno from employee e
EXCEPT
SELECT p.dnum from project p
```

SQL – SET (Küme) Operasyonları – Karşılaştırma (Some)

- Research departmanındaki en az bir çalışandan daha yüksek maaş alan çalışanları listele.

```
SELECT DISTINCT e.fname, e.lname, e.salary
FROM employee AS e, employee AS e2, department AS d
WHERE e.salary > e2.salary AND d.dname = 'Research'
AND e2.dno = d.dnumber;

SELECT e.fname, e.lname, e.salary
FROM employee AS e WHERE e.salary > SOME
(SELECT e2.salary FROM employee AS e2, department AS d
 WHERE d.dname = 'Research' AND e2.dno = d.dnumber);
```

	fname character v	lname character v	salary numeric (10,2)	dname character varying
1	John	Smith	30000.00	Headquarters
2	Alicia	Zelaya	25000.00	Administration
3	Ahmad	Jabbar	25000.00	Administration
4	Jennifer	Wallace	43000.00	Administration
5	James	Borg	55000.00	Research
6	Ramesh	Narayan	38000.00	Research
7	Joyce	English	25000.00	Research
8	Franklin	Wong	40000.00	Research

- > SOME (alt_sorgu) (≡ > ANY) "alt sorgunun döndürdüğü değerlerden en az biri için doğru mu?" diye sorar.

SQL – SET (Küme) Operasyonları – Karşılaştırma (Some)

F <comp> **some** $r \Leftrightarrow \exists\, t \in r$ such that (F <comp> t)
Where <comp> can be: $<, \leq, >, =, \neq$

(5 < **some** | 0 | 5 | 6 |) = true (read: 5 < some tuple in the relation)

(5 < **some** | 0 | 5 |) = false

(5 = **some** | 0 | 5 |) = true

(5 ≠ **some** | 0 | 5 |) = true (since 0 ≠ 5)

(= **some**) ≡ **in**
However, (≠ **some**) ≢ **not in**

SQL – SET (Küme) Operasyonları – Karşılaştırma (All)

- Administration departmanındaki tüm çalışanların maaşından daha yüksek maaş alan çalışanları listeleyiniz.

```
SELECT e.fname, e.lname, e.salary FROM employee AS e
WHERE e.salary > ALL
(SELECT e2.salary FROM employee AS e2, department AS d
WHERE e2.dno = d.dnumber AND d.dname = 'Administration');
```

	fname character v	lname character v	salary numeric (10,2)	dname character varying
1	John	Smith	30000.00	Headquarters
2	Alicia	Zelaya	25000.00	Administration
3	Ahmad	Jabbar	25000.00	Administration
4	Jennifer	Wallace	43000.00	Administration
5	James	Borg	55000.00	Research
6	Ramesh	Narayan	38000.00	Research
7	Joyce	English	25000.00	Research
8	Franklin	Wong	40000.00	Research

SQL – SET (Küme) Operasyonları – Karşılaştırma (All)

F <comp> **all** $r \Leftrightarrow \forall\ t \in r$ (F <comp> *t)*

(5 < **all** | 0 / 5 / 6 |) = false

(5 < **all** | 6 / 10 |) = true

(5 = **all** | 4 / 5 |) = false

(5 ≠ **all** | 4 / 6 |) = true (since 5 ≠ 4 and 5 ≠ 6)

(≠ **all**) ≡ **not in**

However, (= **all**) ≢ **in**

SQL – Form Cümlesindeki Alt Sorgular

- Ortalama maaşı 32.000'in üzerinde olan departmanlarda çalışan personellerin projelerde ki toplam çalışma saatini bulunuz.

```
SELECT SUM(w.hours) AS toplam_calisma_saati
FROM works_on w, employee e WHERE e.ssn = w.essn
AND (SELECT AVG(e2.salary) FROM employee e2 WHERE e2.dno = e.dno) > 32000;
```

SQL – Form Cümlesindeki Alt Sorgular

- SQL, from ifadesinde bir alt sorgu ifadesinin kullanılmasına izin verir.
- Kendi departman ortalamasının üzerinde maaş alan çalışanları bulunuz.

```
SELECT e.fname, e.lname, e.dno, e.salary , da.avg_salary FROM employee e,
  (SELECT e2.dno, AVG(e2.salary) AS avg_salary FROM employee e2 GROUP BY e2.dno) AS da
WHERE da.dno = e.dno AND e.salary > da.avg_salary;
```

	dname character varying	avg numeric
1	Administration	31000.00
2	Headquarters	30000.00
3	Research	39500.00

	fname character v	lname character v	salary numeric (10,2)	dname character varying
1	John	Smith	30000.00	Headquarters
2	Alicia	Zelaya	25000.00	Administration
3	Ahmad	Jabbar	25000.00	Administration
4	Jennifer	Wallace	43000.00	Administration
5	James	Borg	55000.00	Research
6	Ramesh	Narayan	38000.00	Research
7	Joyce	English	25000.00	Research
8	Franklin	Wong	40000.00	Research

SQL – WITH Alt Sorgular

- WITH ifadesi, tanımı yalnızca with ifadesinin bulunduğu sorgu için geçerli olan geçici bir ilişki tanımlamanın bir yolunu sağlar.

- En yüksek maaşlı çalışanı bulunuz.

```
WITH max_salary (value) AS (
   SELECT MAX(salary) FROM employee
)
SELECT e.fname, e.lname, e.salary
FROM employee e, max_salary
WHERE e.salary = max_salary.value;
```

	fname character v	lname character v	salary numeric (10,2)	dname character varying
1	John	Smith	30000.00	Headquarters
2	Alicia	Zelaya	25000.00	Administration
3	Ahmad	Jabbar	25000.00	Administration
4	Jennifer	Wallace	43000.00	Administration
5	James	Borg	55000.00	Research
6	Ramesh	Narayan	38000.00	Research
7	Joyce	English	25000.00	Research
8	Franklin	Wong	40000.00	Research

SQL – WITH Alt Sorgular

- Toplam maaşın tüm departmanlardaki toplam maaş ortalamasından daha yüksek olduğu tüm departmanları bulun

```
WITH dept_totals AS (
  SELECT e.dno, SUM(e.salary) AS total_salary
  FROM employee e
  GROUP BY e.dno
),
overall_avg AS (
  SELECT AVG(total_salary) AS avg_total
  FROM dept_totals
)
SELECT d.dname, dt.total_salary
FROM dept_totals dt, department d, overall_avg oa
WHERE d.dnumber = dt.dno
  AND dt.total_salary > oa.avg_total;
```

	dname character va	total_salary numeric
1	Research	158000.00

SQL – Scalar Alt Sorgular

- Scalar alt sorgu, tek bir değerin beklendiği durumlarda kullanılır.
- Her departman ve o departmandaki çalışanların projelerdeki toplam çalışma saatini bulunuz.

```
SELECT d.dname,
       (SELECT SUM(w.hours)
        FROM works_on w
        WHERE w.essn IN (SELECT e.ssn
                         FROM employee e
                         WHERE e.dno = d.dnumber)) AS total_hours
FROM department d;
```

	dname character varying (15)	total_hours numeric
1	Headquarters	40.0
2	Administration	115.0
3	Research	120.0
4	Math	[null]

- Group by ile yazmayı deneyin!!!

```
SELECT d.dname, SUM(w.hours) AS total_hours FROM department d
LEFT JOIN employee  e ON e.dno  = d.dnumber
LEFT JOIN works_on  w ON w.essn = e.ssn
GROUP BY d.dname;
```

	dname character varying (	total_hours numeric
1	Administration	115.0
2	Headquarters	40.0
3	Research	120.0

SQL – Veritabanının Değiştirilmesi – DELETE

- Tüm çalışanları sil

```
DELETE FROM employee;
```

- "Research" departmanındaki tüm çalışanları sil

```
DELETE FROM employee WHERE dno IN
(SELECT dnumber FROM department WHERE dname = 'Research');
```

- Lokasyonu 'Houston' olan bir departmana bağlı tüm çalışanları sil

```
DELETE FROM employee WHERE dno IN
(SELECT dnumber FROM dept_locations WHERE dlocation = 'Houston');
```

SQL – Veritabanının Değiştirilmesi – DELETE

- Maaşı, tüm çalışanların ortalamasından düşük olan çalışanları sil

```
DELETE FROM employee WHERE salary < (SELECT AVG(salary) FROM employee)
```

- Sorun: Çalışanlar tablosundan kayıtları sildiğimizde ortalama maaş değişir.
- SQL'de kullanılan çözüm: İlk olarak avg (salary)'ı hesaplayın ve silinecek tüm tuple'ları bulun. Sonra, yukarıda bulunan tüm tuple'ları silin (avg'yi yeniden hesaplamadan veya tuple'ları yeniden test etmeden)
- İçteki türetilmiş tablo ortalamayı bir kez hesaplar; silme sırasında değişmez.

```
BEGIN;  -- veya START TRANSACTION;

DELETE FROM employee
WHERE salary < (SELECT AVG(salary) FROM employee)
RETURNING *;    -- silinecek satırları gösterir.

ROLLBACK;       -- değişiklikleri geri al
```

SQL – Veritabanının Değiştirilmesi – INSERT

- 1 nolu departmanda çalışan herkesi 10 nolu projeye 5 saatle ata.

```
INSERT INTO works_on
SELECT e.ssn, 10, 5 FROM employee e WHERE e.dno = 1;
```

	essn [PK] character	pno [PK] inte	hours numeric
1	333445555	10	10.0
2	987987987	10	35.0
3	999887777	10	10.0
	888665555	10	5.0

	ssn [PK] character	fname character v	lname character v	dno integer
1	123456789	John	Smith	5
2	333445555	Franklin	Wong	5
3	453453453	Joyce	English	5
4	666884444	Ramesh	Narayan	5
5	888665555	James	Borg	1
6	987654321	Jennifer	Wallace	4
7	987987987	Ahmad	Jabbar	4
8	999887777	Alicia	Zelaya	4

- INSERT INTO … SELECT … FROM … WHERE … içinde, SELECT-FROM-WHERE kısmı tamamen çalıştırılır ve sonucu sabitlenir; Yani aynı ifadede eklediğiniz yeni satırlar, aynı ifadenin SELECT'inin içine geri düşmez ve seçimi etkilemez.
- Neden önemli? İlk satırı ekler → toplam satır sayısı artar → SELECT bunu tekrar görür → tekrar ekler → sonsuz döngü / patlama olur.

```
INSERT INTO table1 SELECT * FROM table1;
```

SQL – Veritabanının Değiştirilmesi – INSERT

- Research departmanında olup hiçbir projede görev almayan çalışanları 20 nolu projede 8 saatle görevlendir.

```
INSERT INTO works_on (essn, pno, hours)
SELECT e.ssn, 20, 8 FROM employee e
WHERE e.dno = (SELECT dnumber FROM department WHERE dname = 'Research')
  AND NOT EXISTS (SELECT 1 FROM works_on w WHERE w.essn = e.ssn);
```

	ssn character (9)	fname character v	lname character v	dname character varying
1	123456789	John	Smith	Research
2	333445555	Franklin	Wong	Research
3	453453453	Joyce	English	Research
4	666884444	Ramesh	Narayan	Research
5	888665555	James	Borg	Headquarters
6	987654321	Jennifer	Wallace	Administrati...
7	987987987	Ahmad	Jabbar	Administrati...
8	999887777	Alicia	Zelaya	Administrati...

	essn [PK] character (	pno [PK] inte	hours numeric
1	123456789	1	32.5
2	333445555	2	10.0
3	453453453	1	20.0
4	666884444	3	40.0
5	987654321	20	15.0
6	987987987	10	35.0
7	999887777	10	10.0
	888665555	20	8.0

SQL – Veritabanının Değiştirilmesi – UPDATE

- Tüm çalışanlara %5 zam

```
UPDATE employee SET salary = salary * 1.05;
```

- Maaşı 70.000'den düşük olanlara %5 zam

```
UPDATE employee SET salary = salary * 1.05 WHERE salary < 35000;
```

- Maaşı genel ortalamanın altında olanlara %5 zam

```
UPDATE employee SET salary = salary * 1.05
WHERE salary < (SELECT AVG(salary) FROM employee);
```

SQL – Veritabanının Değiştirilmesi – UPDATE

- Maaşı kendi departman ortalamasının altında olanlara %5 zam

```
UPDATE employee e SET salary = e.salary * 1.05
WHERE e.salary < (SELECT AVG(e2.salary) FROM employee e2
  WHERE e2.dno = e.dno);
```

- Maaşı 40.000 üzerinde olan çalışanların maaşlarını %3, diğerlerini ise %5 artırın.

```
UPDATE employee
SET salary = salary * 1.03
WHERE salary > 40000;

UPDATE employee
SET salary = salary * 1.05
WHERE salary <= 40000;
```

```
UPDATE employee
SET salary = salary * CASE
  WHEN salary > 40000 THEN 1.03
  ELSE 1.05
END;
```

- Sıralama önemlidir.

SQL – Veritabanının Değiştirilmesi – UPDATE

- Her çalışanın projelerde toplam çalışma süresini «sum_proj_hours» sütununa yaz.

```
UPDATE employee e
SET sum_proj_hours = (SELECT SUM(w.hours)
FROM works_on w WHERE w.essn = e.ssn);
```

	ssn [PK] character	fname character v	sum_proj_hours character varying
1	123456789	John	40.0
2	333445555	Franklin	40.0
3	453453453	Joyce	40.0
4	666884444	Ramesh	0
5	888665555	James	13.0
6	987654321	Jennifer	35.0
7	987987987	Ahmad	40.0
8	999887777	Alicia	40.0

- Herhangi bir projede görev almamış personel için «proj_count» değerini null olarak ayarlar.

- sum(w.hours) yerine şunu kullanın:

```
UPDATE employee e
SET sum_proj_hours = (SELECT CASE WHEN SUM(w.hours)
IS NOT NULL THEN SUM(w.hours)ELSE 0 END
FROM works_on w WHERE w.essn = e.ssn);
```

YILDIZ TEKNİK ÜNİVERSİTESİ

Yasemin Topuz

Yıldız Teknik Üniversitesi

ytopuz@yildiz.edu.tr