



Veri Tabanı Yönetim Sistemleri

Bölüm 5–Alt Sorgular (SubQueries)

Dr. Öğr. Üyesi Elvan DUMAN

Bölüm 5

Alt Sorgular (SubQueries)

Bölüm 5 - Amaçlar



- ▶ Alt Sorguların Temel Kavramları, Kullanım Alanları
- ▶ Alt Sorguların Performans Etkileri
- ▶ ANY, ALL, IN, EXISTS ve NOT EXISTS Gibi Operatörlerle Alt Sorgular
- ▶ UNION, INTERSECT VE EXCEPT Alt Sorguları
- ▶ Birleştirme (JOIN) Uygulamaları
- ▶ Ekleme, Çıkarma Silme İşlemleri (INSERT, UPDATE, DELETE)

SQL'de Alt Sorgular: Tanım ve Kullanım



- ▶ Alt sorgular, başka bir sorgunun içinde çalıştırılan iç içe SELECT ifadeleridir.
- ▶ Tek bir değer döndüren (scalar), çoklu satır döndüren (multiple-row) ve tablo döndüren (table) alt sorgular olabilir.
- ▶ WHERE, HAVING, FROM ve SELECT ifadelerinde kullanılabilirler.
- ▶ SELECT deyimi içinde kullanılabileceği gibi, INSERT, UPDATE ve DELETE deyimlerinde de görülebilir.
- ▶ **Ana sorgudan önce çalışır ve ana sorguya veri sağlar.**
- ▶ **Örnek Kullanım:** sorgu, maaşı tüm personelin ortalama maaşından yüksek olan çalışanları getirir.

```
SELECT fName, lName, salary  
FROM Staff  
WHERE salary > (SELECT AVG(salary) FROM Staff);
```

SQL'de Alt Sorgular: Tanım ve Kullanım



Alt sorgular, ana sorgunun bir parçası olarak çalışır ve belirli kriterlere göre veri çeker. Bir alt sorgunun **tek bir değer döndürmesi durumunda**, =` operatörü ile kullanılabilir.

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo =
  (SELECT branchNo
   FROM Branch
   WHERE street = '163 Main St');
```

İç sorgu sonrası

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo = 'B003';
```

staffNo	fName	lName	position
SG37	Ann	Beech	Assistant
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

Bu sorgu, '163 Main St' adresindeki şubede çalışan personeli listeler.

Alt sorgu, '163 Main St' adresine sahip şubenin branchNo değerini döndürür. Ana sorgu, bu branchNo değerine eşit olan personelleri seçer.

Küme ile Alt Sorgu



Alt sorgular, birden fazla değer döndürdüğünde küme fonksiyonlarıyla birlikte kullanılabilir.

Örneğin, ortalama maaştan daha fazla maaş alan tüm personeli ve ortalamaya göre ne kadar fazla aldığını listeleyin.

```
SELECT staffNo, fName, lName, position,  
       salary - (SELECT AVG(salary) FROM Staff) AS SalDiff  
FROM Staff  
WHERE salary > (SELECT AVG(salary) FROM Staff);
```

staffNo	fName	lName	position	salDiff
SL21	John	White	Manager	13000.00
SG14	David	Ford	Supervisor	1000.00
SG5	Susan	Brand	Manager	7000.00

Küme ile Alt Sorgu



SQL'de grup fonksiyonları (AVG, SUM, COUNT, vb.) doğrudan WHERE içinde kullanılamaz

- Bir önceki sorguda : **WHERE salary > AVG(salary);** -- *ŞEKLİNDE OLSA HATA OLURDU*

Ancak eğer ortalama maaş sabit bir değer olarak biliniyorsa (örneğin, 17000), sorgu şu şekilde yazılabilir:

```
SELECT staffNo, fName, lName, position,  
       salary - 17000 AS salDiff  
FROM Staff  
WHERE salary > 17000;
```

- Ancak bu yöntem dinamik değildir ve ortalama maaş değiştiğinde güncellenmesi gerekir. Alt sorgu kullanmak daha sağlıklıdır.

Alt Sorgu Kuralları



Kural	Açıklama
ORDER BY kullanımı	ORDER BY, alt sorgularda kullanılamaz, yalnızca ana sorguda kullanılabilir.
SELECT listesi tek sütun içermeli	EXISTS dışında kullanılan alt sorgular, tek bir sütun döndürmelidir.
Varsayılan sütun isimlendirmesi	Sütunlar, alt sorgunun FROM cümleciğindeki tablo adına göre adlandırılır.
Takma ad (Alias) kullanımı	FROM içindeki tablolar takma ad (alias) ile adlandırılabilir.

Alt Sorgu Kuralları



- ▶ Bir SQL sorgusunda alt sorgu, bir karşılaştırma işlemi (**örneğin, = , <, >, <=, >=, <>**) içerdiğinde, alt sorgu karşılaştırmının sağ tarafında bulunmalıdır.
- ▶ Örneğin, aşağıdaki kullanım doğrudur:

```
SELECT staffNo, fName, lName, position  
FROM Staff  
WHERE salary > (SELECT AVG(salary) FROM Staff);
```

Ancak **hatalı bir kullanım** şu şekilde olurdu:

```
SELECT staffNo, fName, lName, position  
FROM Staff  
WHERE (SELECT AVG(salary) FROM Staff) < salary;
```

Alt Sorgu : IN Kullanımı



- ▶ IN operatörü, bir değerin bir alt sorgudan dönen sonuç kümesinde olup olmadığını kontrol etmek için kullanılır.
- ▶ **Tek bir değere değil, birden fazla değere karşılık gelen sonuçları** döndüren alt sorgular ile kullanılır. Yani Birden fazla değeri eşleştirme gerektiren durumlarda = operatörü yerine IN operatörü tercih edilir.

'163 Main St' şubesinde çalışan personellerin yönetiminde olan tüm mülkleri listeleyiniz.

```
▶ SELECT propertyNo, street, city, postcode, type, rooms, rent
FROM PropertyForRent
WHERE staffNo IN
  (SELECT staffNo
   FROM Staff
   WHERE branchNo =
    (SELECT branchNo
     FROM Branch
     WHERE street = '163 Main St'));
```

propertyNo	street	city	postcode	type	rooms	rent
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375
PG21	18 Dale Rd	Glasgow	G12	House	5	600

Alt Sorgu : ANY ve ALL Kullanımı



- ▶ ANY ve ALL operatörleri, alt sorgularla kullanılarak çoklu karşılaştırmalar yapmayı sağlar.
 - **ANY:** Alt sorgudan dönen herhangi bir değerle koşulu sağlayan satırları seçer.
 - **ALL:** Alt sorgudan dönen tüm değerlerle koşulu sağlayan satırları seçer.
- ▶ Eğer alt sorgu boş dönüyorsa
 - **ALL:** true döndürür (çünkü hiçbir değer karşılaştırmaya aykırı değildir).
 - **ANY:** false döndürür (çünkü karşılaştırılacak hiçbir değer yoktur).

Örnek Kullanım: Maaşı B003 şubesindeki tüm personellerin maaşlarından büyük olan personelleri listele:

```
SELECT staffNo, fName, lName, salary
FROM Staff
WHERE salary > ALL (SELECT salary FROM Staff WHERE branchNo = 'B003');
```

Alt Sorgu : ANY ve ALL Kullanımı



Örnek Kullanım: Maaşı B003 şubesindeki tüm personellerin maaşlarından büyük olan personelleri listele:

```
SELECT staffNo, fName, lName, salary  
FROM Staff  
WHERE salary > ALL (SELECT salary FROM Staff WHERE branchNo = 'B003');
```

- İç sorgu {12000, 18000, 24000} kümesini üretir ve dıştaki sorgu maaşları bu kümedeki değerlerin tümünden büyük maaşı olan personeller listelenir. Küme boş gelseydi herkes listelenirdi.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00

Alt Sorgu : ANY ve ALL Kullanımı



Örnek Kullanım: Maaşı B003 şubesindeki en az bir personelin maaşından büyük olan personelleri listele:

SELECT staffNo, fName, lName, salary

FROM Staff

WHERE salary > **ANY** (**SELECT** salary **FROM** Staff **WHERE** branchNo = 'B003');

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

Çoklu Tablo Sorguları



- ▶ Sonuç sütunları aynı tablodan geldiğinde, alt sorgular doğrudan kullanılabilir.
- ▶ Sonuç sütunları birden fazla tablodan geldiğinde, bir birleşim (JOIN) kullanmak gereklidir.
- ▶ Birleşimi gerçekleştirmek için, FROM cümleciğinde birden fazla tablo belirtilmelidir.
- ▶ Tabloların bağlanması, virgül (,) ile ayrılır ve WHERE cümleciği ile ilişkilendirilir.

Örnek Kullanım: Londra şubelerinde çalışan personelleri listeleme

```
SELECT Staff.staffNo, Staff.fName, Staff.lName, Branch.city  
FROM Staff, Branch  
WHERE Staff.branchNo = Branch.branchNo  
      AND Branch.city = 'London';
```

staffNo	fName	lName	city
SL21	John	White	London
SL41	Julie	Lee	London

Çoklu Tablo Sorgularında Takma Ad (Alias) Kullanımı



- ▶ FROM cümleciğinde tablo için bir takma ad (alias) kullanılabilir.
- ▶ Takma ad, tablo isminden bir boşluk ile ayrılır.
- ▶ Takma adlar, belirsizlik olduğunda sütun adlarını nitelemek için kullanılır.

Örnek Kullanım

```
SELECT Staff.staffNo, Staff.fName,  
Staff.lName, Branch.city  
FROM Staff, Branch  
WHERE Staff.branchNo = Branch.branchNo  
AND Branch.city = 'London';
```

```
SELECT S.staffNo, S.fName, S.lName, B.city  
FROM Staff AS S, Branch AS B  
WHERE S.branchNo = B.branchNo  
AND B.city = 'London';
```

Basit Birleştirme



- ▶ Verilen herhangi bir açıklama ile birlikte bir emlak inceleyen tüm müşterileri listeleyin.
- ▶ **SELECT** c.clientNo, fName, lName, propertyNo, comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;

clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

- ▶ WHERE c.clientNo = v.clientNo; koşulu, yalnızca iki tabloda eşleşen müşteri numaralarını içeren satırları döndürmesini sağlar.
- ▶ Bu yöntem, ilişkisel cebirdeki eşit birleştirme (Equi-Join) işlemine eşdeğerdir.

Basit Birleştirme



- Her şube için, emlakları yöneten personeli, şubenin bulunduğu şehri ve yönettikleri emlakları listeleyin.

```
SELECT b.branchNo, b.city, s.staffNo, fName, lName, propertyNo
FROM Branch b, Staff s, PropertyForRent p
WHERE b.branchNo = s.branchNo AND s.staffNo = p.staffNo
ORDER BY b.branchNo, s.staffNo, propertyNo;
```

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

Birleştirme (JOIN)



- ▶ SQL'de tabloları birleştirmenin birden fazla yolu vardır. WHERE kullanarak alternatif birleştirmeyi bir önceki slaytta gördük **ancak, modern SQL'de JOIN kullanımı daha okunaklı ve indeksleme daha iyi çalıştığından performans açısından daha iyidir.**
- ▶ Join, veri tutarlılığını daha iyi korur ve hatalı eşleşmeleri önler.
- ▶ **Örnek Kullanım:** Verilen herhangi bir açıklama ile birlikte bir emlak inceleyen tüm müşterileri listeleyin.
- ▶ **SELECT** c.clientNo, c.fName, c.lName, v.propertyNo, v.comment
FROM Client c
INNER JOIN Viewing v **ON** c.clientNo = v.clientNo;

Birleştirme (JOIN)



- Her şube için, emlakları yöneten personeli, şubenin bulunduğu şehri ve yönettikleri emlakları listeleyin.
- **SELECT** b.branchNo, b.city, s.staffNo, s.fName, s.lName, p.propertyNo
FROM Branch b
INNER JOIN Staff s **ON** b.branchNo = s.branchNo
INNER JOIN PropertyForRent p **ON** s.staffNo = p.staffNo
ORDER BY b.branchNo, s.staffNo, p.propertyNo;

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

Birleştirme – LEFT JOIN



- ▶ Tüm şubeleri ve aynı şehirdeki kiralık emlakları listeleyin
- ▶ Eğer bir şube var ve o şehirde emlak yoksa şube kaybolmaz karşısında null değer gelir.
- ▶ **SELECT** b.branchNo, b.city, p.propertyNo, p.street, p.type, p.rent
FROM Branch b
LEFT JOIN PropertyForRent p **ON** b.city = p.city
ORDER BY b.city, b.branchNo, p.propertyNo;

branchNo	city ▲ 1	propertyNo	street	type	rent
B007	Aberdeen	PA14	16 Holhead	House	650
B004	Bristol	NULL	NULL	NULL	NULL
B003	Glasgow	PG16	5 Novar Dr	Flat	450
B003	Glasgow	PG21	18 Dale Rd	House	600
B003	Glasgow	PG36	2 Manor Rd	Flat	375
B003	Glasgow	PG4	6 Lawrence St	Flat	350
B002	London	PL94	6 Argyll St	Flat	400
B005	London	PL94	6 Argyll St	Flat	400

Birleştirme – RIGHT JOIN



- ▶ Tüm emlakları ve aynı şehirdeki şubeleri listeleyin
- ▶ Eğer bir emlak var ve o şehirde şube yoksa emlak kaybolmaz karşısında null değer gelir.
- ▶ **SELECT** b.branchNo, b.city, p.propertyNo, p.street, p.type, p.rent
FROM Branch b
RIGHT JOIN PropertyForRent p **ON** b.city = p.city
ORDER BY b.city, b.branchNo, p.propertyNo;

branchNo	city	propertyNo	street	type	rent
NULL	NULL	PG4	6 Lawrence St	Flat	350
B007	Aberdeen	PA14	16 Holhead	House	650
B003	Glasgow	PG16	5 Novar Dr	Flat	450
B003	Glasgow	PG21	18 Dale Rd	House	600
B003	Glasgow	PG36	2 Manor Rd	Flat	375
B002	London	PL94	6 Argyll St	Flat	400
B005	London	PL94	6 Argyll St	Flat	400

Birleştirme – FULL OUTER JOIN



- ▶ Bu sorgu, Branch (Şube) ve PropertyForRent (Kiralık Emlaklar) tablolarını FULL OUTER JOIN ile birleştirerek,
 - Aynı şehirde bulunan şubeleri ve emlakları listeler.
 - Herhangi bir emlakla eşleşmeyen şubeleri de içerir. (LEFT JOIN özelliklerini taşır)
 - Herhangi bir şubeye bağlı olmayan emlakları gösterir. (RIGHT JOIN özelliklerini taşır)
- ▶ **SELECT** b.branchNo, b.city, p.propertyNo, p.street, p.type, p.rent
FROM Branch b
FULL OUTER JOIN PropertyForRent p **ON** b.city = p.city
ORDER BY b.city, b.branchNo, p.propertyNo;
- ▶ MySQL'de FULL OUTER JOIN doğrudan desteklenmez! FULL OUTER JOIN yerine LEFT ve RIGHT JOIN işlemiyle UNION kullanarak aynı sonucu almak mümkündür.

EXISTS ve NOT EXISTS



- ▶ Bir koşulun var olup olmadığını kontrol eder ve alt sorgu tarafından en az bir satır döndürülüyorsa TRUE veya FALSE sonucu üretir.
- ▶ EXISTS ve NOT EXISTS sadece alt sorgular ile kullanılır.
- ▶ Alt sorgu boş bir sonuç döndürüyorsa FALSE (Yanlış) döndürür.
- ▶ NOT EXISTS, EXISTS'in tersidir, yani alt sorgu boşsa TRUE, doluysa FALSE döndürür.

▶ **Örnek Kullanım:** En az bir müşterinin incelediği mülkleri listeleyelim.

```
SELECT propertyNo, street, city
FROM PropertyForRent p
WHERE EXISTS (
    SELECT *
    FROM Viewing v
    WHERE p.propertyNo = v.propertyNo
);
```

propertyNo	street	city
PA14	16 Holhead	Aberdeen
PG36	2 Manor Rd	Glasgow
PG4	6 Lawrence St	Glasgow

EXISTS ve NOT EXISTS



- ▶ **NOT EXISTS Örneği:** Hiçbir müşteri tarafından incelenmemiş mülkleri listeleyelim:
- ▶ **SELECT** propertyNo, street, city
FROM PropertyForRent p
WHERE NOT EXISTS (
 SELECT *
 FROM Viewing v
 WHERE p.propertyNo = v.propertyNo
);

propertyNo	street	city
PG16	5 Novar Dr	Glasgow
PG21	18 Dale Rd	Glasgow
PL94	6 Argyll St	London

EXISTS ve NOT EXISTS



- Bir Londra şubesinde çalışan personeli listeleyiniz.
- **SELECT** staffNo, fName, lName, position
FROM Staff s
WHERE EXISTS
 (**SELECT** *
 FROM Branch b
 WHERE s.branchNo = b.branchNo **AND**
 city = 'London');

staffNo	fName	lName	position
SL21	John	White	Manager
SL41	Julie	Lee	Assistant

UNION, INTERSECT VE DIFFERENCE (EXCEPT)

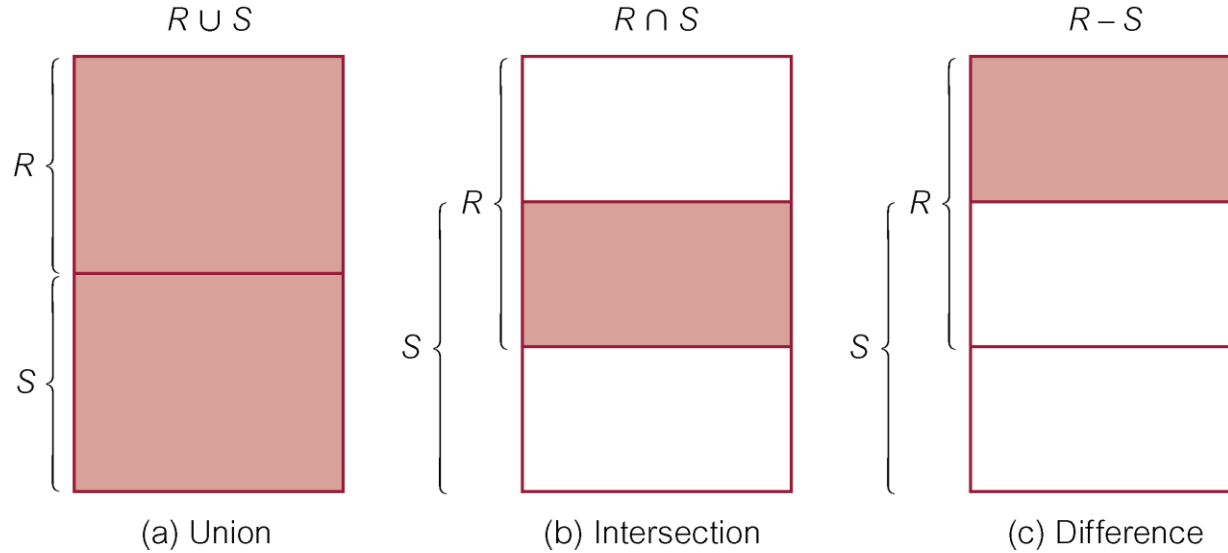


- SQL'de iki veya daha fazla sorgunun sonuçlarını tek bir tabloda birleştirmek için küme işlemleri kullanılır.

Küme İşlemi	Açıklama
UNION	A veya B veya her ikisinde de bulunan tüm satırları içeren bir sonuç döndürür. Tüm kayıtları birleştirir, tekrar edenleri kaldırır.
INTERSECT	Hem A hem de B'de ortak olan satırları içeren bir sonuç döndürür.
EXCEPT (DIFFERENCE)	A'da olup B'de olmayan satırları içeren bir sonuç döndürür.

- Her iki sorgunun da sütun sayısı ve veri türleri uyumlu olmalıdır.
- **MySQL'de INTERSECT ve EXCEPT 2022 yılında yayımlanan (v8.0.31) sürümü ile desteklenmiştir.**

UNION, INTERSECT VE DIFFERENCE (EXCEPT)



UNION, INTERSECT VE DIFFERENCE (EXCEPT)



► UNION Kullanımı

► Bir şube veya emlağı olan tüm şehirleri listeleyin

```
► (SELECT city  
   FROM Branch  
   WHERE city IS NOT NULL)  
UNION  
(SELECT city  
   FROM PropertyForRent  
   WHERE city IS NOT NULL);
```

city
London
Glasgow
Aberdeen
Bristol

► UNION operatörü, iki alt sorgudan gelen sonuçları birleştirir.

► Tekrarlanan şehir isimlerini kaldırarak benzersiz (distinct) bir liste oluşturur.

UNION, INTERSECT VE DIFFERENCE (EXCEPT)



► INTERSECT Kullanımı

► Hem bir şubesi ve hem de kiralık bir emlakı olan tüm şehirleri listeleyin.

► **(SELECT city FROM Branch)**
INTERSECT
(SELECT city FROM PropertyForRent);

► INTERSECT kullanmadan bu sorgu şu şekilde yazılabilir:

SELECT DISTINCT city FROM Branch b
WHERE EXISTS
(SELECT * FROM PropertyForRent p
WHERE p.city = b.city);

city
Aberdeen
Glasgow
London

UNION, INTERSECT VE DIFFERENCE (EXCEPT)



- ▶ **EXCEPT Kullanımı**

- ▶ Bir şube ofisi olan ama hiçbir emlağı olmayan tüm şehirlerin listesi.

- ▶ **(SELECT city FROM Branch)**
EXCEPT
(SELECT city FROM PropertyForRent);

city
Bristol

- ▶ EXCEPT kullanmadan bu sorgu farklı şekillerde yazılabilir:

```
SELECT DISTINCT city FROM Branch
WHERE city NOT IN
  (SELECT city FROM PropertyForRent);
```

```
SELECT DISTINCT city FROM Branch b
WHERE NOT EXISTS
  (SELECT * FROM PropertyForRent p
   WHERE p.city = b.city);
```

INSERT, UPDATE, DELETE



► INSERT Kullanımı

- Yeni bir kayıt eklemek için kullanılır.

Tek bir satır ekleme

- `INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);`
- `INSERT INTO Client (clientNo, fName, lName, telNo, prefType, maxRent) VALUES ('C100', 'Ahmet', 'Yılmaz', '0212-1234567', 'Flat', 500);`

INSERT, UPDATE, DELETE



► INSERT Kullanımı

- Yeni bir kayıt eklemek için kullanılır.

Çoklu satır ekleme: veri virgül ile ayrılır

- **INSERT INTO** Branch (**branchNo**, **street**, **city**, **postcode**)
VALUES
('B006', '45 Elm St', 'Manchester', 'M1 1AA'),
('B007', '67 Oak Rd', 'Liverpool', 'L2 3BB');

INSERT, UPDATE, DELETE



► UPDATE Kullanımı

- Mevcut kayıtların güncellenmesinde kullanılır

- UPDATE *table_name*

SET *column1* = *value1*, *column2* = *value2*, ...
WHERE *condition*;

- UPDATE Staff

SET salary = salary * 1.10
WHERE staffNo = 'SG37';

- UPDATE PropertyForRent

SET rent = rent + 50
WHERE type = 'Flat';

INSERT, UPDATE, DELETE



▶ DELETE Kullanımı

▶ Mevcut kayıtların silinmesinde kullanılır

▶ **DELETE FROM** table_name **WHERE** condition;

▶ **DELETE FROM** Client
WHERE clientNo = 'C100';

▶

▶ **DELETE FROM** Viewing
WHERE comment = 'too small';

Bölüm sonu- Teşekkürler.

Dr. Öğr. Üyesi Elvan DUMAN – eduman@mehmetakif.edu.tr