

# Veritabanı Yönetim Sistemleri

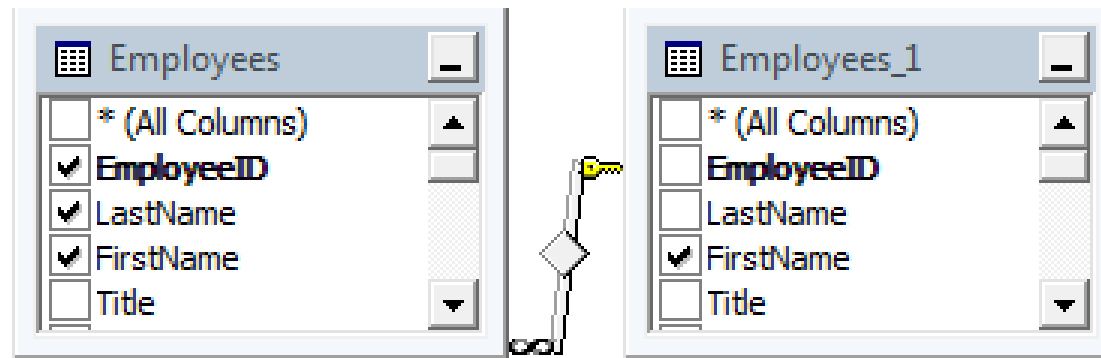
## İleri SQL



# Konular

- ✓ Özyineli Sorgular
- ✓ Karmaşık Sorgular
- ✓ Ürün Sipariş Sistemi
- ✓ DML ile Alt Sorgu Kullanımı
- ✓ Where ile Alt Sorgu (Tek Değer Döndüren) Kullanımı
- ✓ Where ile Alt Sorgu (Çok Değer Döndüren) Kullanımı
- ✓ Having ile Alt Sorgu Kullanımı
- ✓ From ile Alt Sorgu Kullanımı
- ✓ Inline Alt Sorgu Kullanımı
- ✓ İlintili (Correlated) Alt Sorgu Kullanımı

# Özyineli Join



```
SELECT  dbo.Employees.EmployeeID, dbo.Employees.LastName, dbo.Employees.FirstName, Employees_1.FirstName AS Expr1
FROM    dbo.Employees INNER JOIN Employees_1
      ON dbo.Employees.ReportsTo = Employees_1.EmployeeID
```

	EmployeeID	LastName	FirstName	Expr 1
▶	1	Davolio	Nancy	Andrew
	3	Leverling	Janet	Andrew
	4	Peacock	Margaret	Andrew
	5	Buchanan	Steven	Andrew
	6	Suyama	Michael	Steven
	7	King	Robert	Steven

## Karmaşık Sorgular

Karmaşık sorgu bir sorgu içerisinde birden çok SELECT deyiminin kullanılması işlemi olarak tarif edilir. Karmaşık sorguda ana sorgu bir başka alt sorgudan elde edilen sonuçları kullanır.

### Alt sorgu nedir?

Bazı durumlarda bir sorgudan elde edilen sonuç , bir diğer sorguyu ilgilendirebilir. Bu durumda alt sorgu ya da başka bir isimle iç sorgular kullanılır.

Örnek: Hangi personelin ücreti “Ahmet” isimli personelin ücretinden daha fazladır. Bu ifadeyi iki farklı sorguya ayırmak mümkündür:

1. “Ahmet” isimli personelin aylık ücreti
2. Hangi personelin ücreti “Ahmet” isimli personelin ücretinden daha fazladır.

**Ana sorgu:** Hangi personel “Ahmet” in ücretinden daha fazla alır?

**Alt sorgu:** “Ahmet” in aylık ücreti nedir?

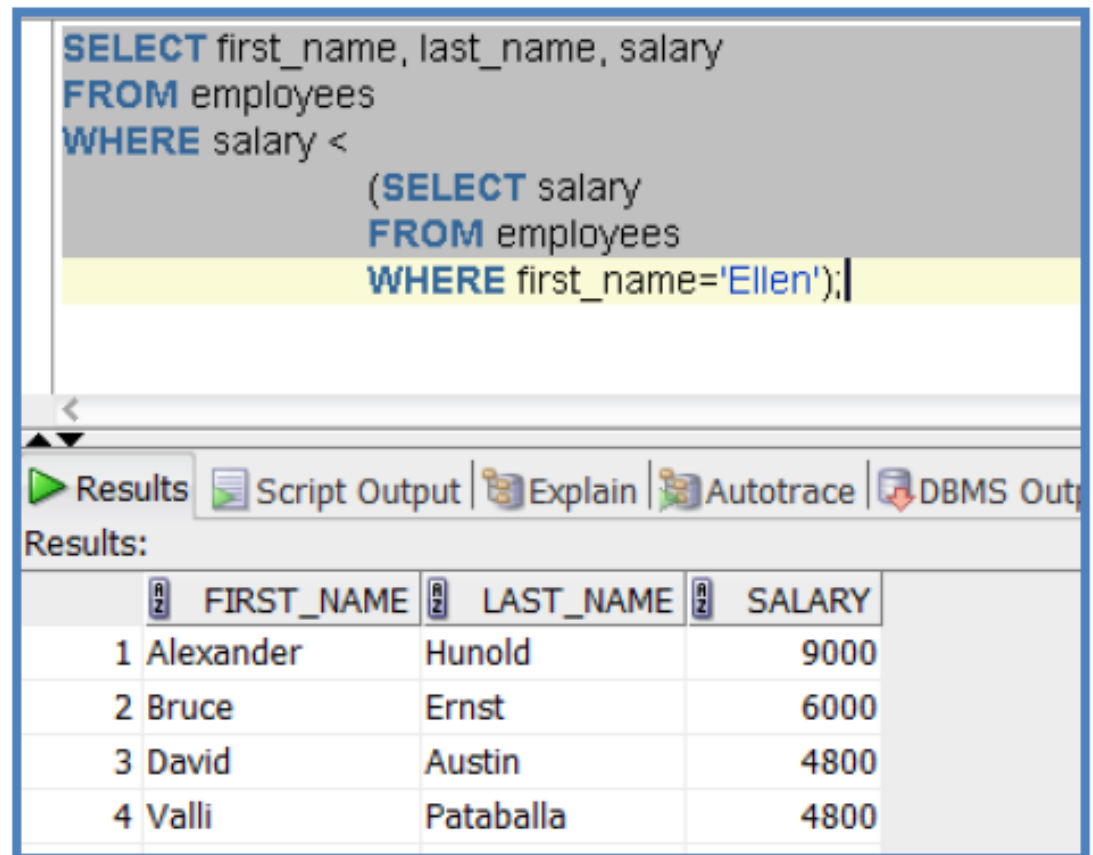
```
SELECT ADI, GÖREVİ, ÜCRET  
FROM PERSONEL  
WHERE ÜCRET >  
      (SELECT ÜCRET  
       FROM PERSONEL  
       WHERE ADI='Ahmet');
```

## Alt Sorgu Düzenleme Kuralları

1. Alt sorgu ana sorgunun sağında yer almalıdır. FROM sözcüğü bu kuralın dışındadır.
2. Alt sorgu ( ) parentez işaretleri arasında yer alır.
3. Alt sorguda ORDER By sözcüğü kullanılmaz. Bu sözcük ana sorgu içerisinde yer alabilir.
4. Alt sorgularda tek satır ve çok satır olmak üzere iki tür karşılaştırma operatörleri kullanılabilir.

### Tek Satır Sorguları

Sadece bir değer döndüren sorgulardır. İç taraftaki SELECT deyiminden sadece tek değer elde edilir. Bu sorgularda aşağıdaki operatörler kullanılır.



```
SELECT first_name, last_name, salary
FROM employees
WHERE salary <
      (SELECT salary
       FROM employees
       WHERE first_name='Ellen');
```

Results: Script Output Explain Autotrace DBMS Out

	FIRST_NAME	LAST_NAME	SALARY
1	Alexander	Hunold	9000
2	Bruce	Ernst	6000
3	David	Austin	4800
4	Valli	Pataballa	4800

# GRUP FONKSİYONLARININ KULLANILMASI

Grup fonksiyonları sadece bir değer döndürdükleri için, bu tür fonksiyonlar tek satır alt sorgulama işlemlerinde kullanılabilir.

The screenshot shows a SQL query editor with the following text:

```
SELECT first_name, last_name, job_id, salary
FROM employees
WHERE salary =
      (SELECT MAX(salary)
       FROM employees);
```

Below the query editor, there is a toolbar with icons for Results, Script Output, Explain, Autotrace, DBMS Output, and OWB. The 'Results' tab is selected, and the results are displayed in a table format.

Results:

	FIRST_NAME	LAST_NAME	JOB_ID	SALARY
1	Steven	King	AD_PRES	24000

## HAVING İLE GRUP ŞARTLARININ TANIMLANMASI

SELECT liste  
FROM tablo  
GROUP BY sütun  
HAVING fonksiyon, operatör (işleç)  
(SELECT fonksiyon  
FROM tablo);

```
SELECT department_id, MAX(salary)
FROM employees
GROUP BY department_id
HAVING MAX(salary) >
      (SELECT AVG(salary)
       FROM employees
       WHERE department_id = 30);
```

Results | Script Output | Explain | Autotrace | DBMS

Results:

	DEPARTMENT_ID	MAX(SALARY)
1	100	12000
2	30	11000
3	(null)	7000
4	90	24000
5	20	13000
6	70	10000

# ÇOKLU SATIR ALT SORGUSU

Alt sorgudan bir Satır yerine daha fazla sayıda satır elde ediliyorsa bu kez sorgu çoklu satır alt sorgusu olarak değerlendirilir. Bu tür sorgular aşağıdaki operatörler yardımıyla gerçekleştirilir.

**IN** liste içerisindeki değerlerden seçme işlemini gerçekleştirir

**ANY** alt sorgu tarafından döndürülen her değer için karşılaştırma işlemini gerçekleştirir.

**ALL** alt sorgu tarafından döndürülen tüm değerlerle karşılaştırma işlemini gerçekleştirir.



### ÇOKLU SORGUDA IN

```
SELECT first_name, last_name, salary, department_id
FROM employees
WHERE salary IN (SELECT MAX(salary)
                  FROM employees
                  GROUP BY department_id);
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA

Results:

	FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_ID
1	Steven	King	24000	90
2	Alexander	Hunold	9000	60
3	Nancy	Greenberg	12000	100
4	Daniel	Faviet	9000	100
5	John	Chen	8200	100
6	Den	Raphaely	11000	30

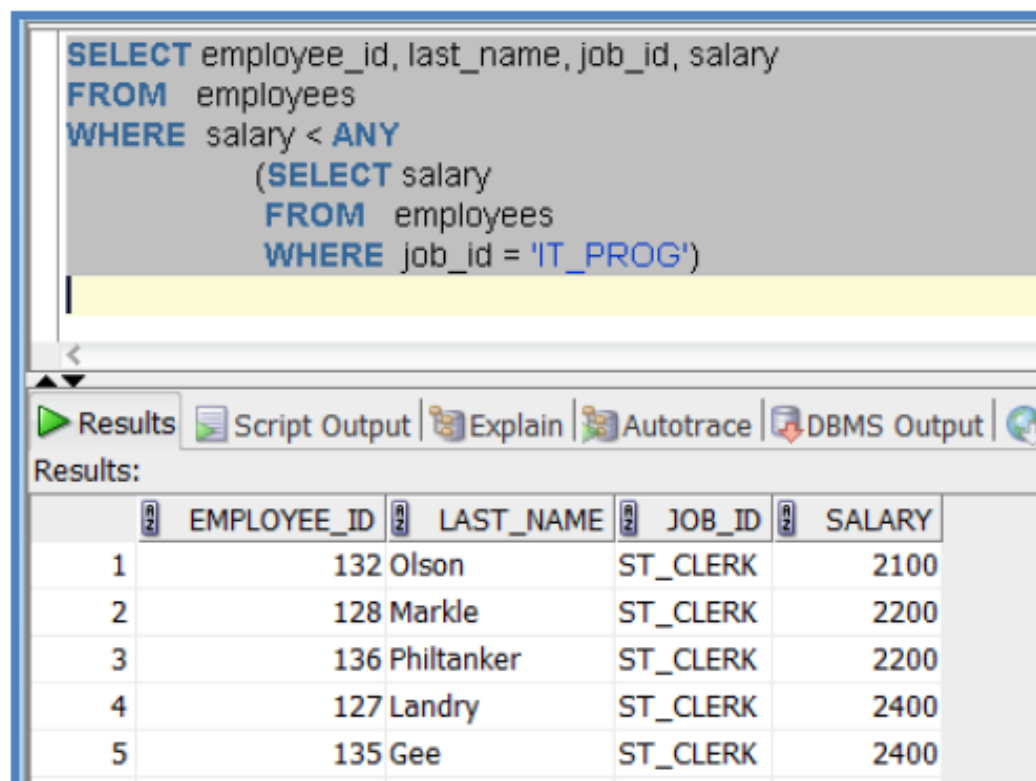
## ÇOKLU SORGUDA ANY

Alt sorgu tarafından üretilen her bir değer ana sorgu içinde belirlenen bir değerle karşılaştırmak söz konusu ise ANY operatörü kullanılır. Bu operatör “=, <, >” operatörlerinden biri ile kullanılabilir.

=ANY ifadesi IN tanımı ile eşdeğerdir.

<ANY en çoktan daha az anlamına gelir

>ANY en az dan daha büyük anlamına gelir.



```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ANY
      (SELECT salary
       FROM employees
       WHERE job_id = 'IT_PROG')
```

Results: Script Output Explain Autotrace DBMS Output

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	132	Olson	ST_CLERK	2100
2	128	Markle	ST_CLERK	2200
3	136	Philtanker	ST_CLERK	2200
4	127	Landry	ST_CLERK	2400
5	135	Gee	ST_CLERK	2400

## ANY / SOME

```
SELECT empno, sal
FROM   emp
WHERE  sal > ANY (2000, 3000, 4000);
```

EMPNO	SAL
7566	2975
7698	2850
7782	2450
7788	3000
7839	5000
7902	3000

SQL>

-- Transformed to equivalent statement without ANY.

```
SELECT empno, sal
FROM   emp
WHERE  sal > 2000 OR sal > 3000 OR sal > 4000;
```

```
SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  e1.sal > ANY (SELECT e2.sal
                     FROM   emp e2
                     WHERE  e2.deptno = 10);
```

## ÇOKLU SORGUDA ALL

Alt sorgu tarafından üretilen tüm değerlerin, ana sorgu içerisinde bir girdi olarak kullanılması söz konusu ise ALL operatörü kullanılır.

<ALL en küçükten daha küçük anlamına gelir

>ALL en büyüktten daha büyük anlamına gelir.

```
SELECT employee_id, first_name, last_name, job_id, salary
FROM employees
WHERE salary < ALL
      (SELECT salary
       FROM employees
       WHERE job_id = 'SA_MAN')
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	JOB_ID	SALARY
1	156	Janette	King	SA_REP	10000
2	150	Peter	Tucker	SA_REP	10000
3	204	Hermann	Baer	PR_REP	10000
4	169	Harrison	Bloom	SA_REP	10000
5	170	Taylor	Fox	SA_REP	9600
6	163	Danielle	Greene	SA_REP	9500

```
SELECT * FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-1980 00:00:00	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-1981 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-1981 00:00:00	1250	500	30
7566	JONES	MANAGER	7839	02-APR-1981 00:00:00	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-1981 00:00:00	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-1981 00:00:00	2850		30
7782	CLARK	MANAGER	7839	09-JUN-1981 00:00:00	2450		10
7788	SCOTT	ANALYST	7566	19-APR-1987 00:00:00	3000		20
7839	KING	PRESIDENT		17-NOV-1981 00:00:00	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-1981 00:00:00	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-1987 00:00:00	1100		20
7900	JAMES	CLERK	7698	03-DEC-1981 00:00:00	950		30
7902	FORD	ANALYST	7566	03-DEC-1981 00:00:00	3000		20
7934	MILLER	CLERK	7782	23-JAN-1982 00:00:00	1300		10

```
SELECT empno, sal
FROM emp
WHERE sal > ALL (2000, 3000, 4000);
```

EMPNO	SAL
7839	5000

```
SELECT empno, sal
FROM emp
WHERE sal > 2000 AND sal > 3000 AND sal > 4000;
```

```
SELECT * FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-1980 00:00:00	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-1981 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-1981 00:00:00	1250	500	30
7566	JONES	MANAGER	7839	02-APR-1981 00:00:00	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-1981 00:00:00	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-1981 00:00:00	2850		30
7782	CLARK	MANAGER	7839	09-JUN-1981 00:00:00	2450		10
7788	SCOTT	ANALYST	7566	19-APR-1987 00:00:00	3000		20
7839	KING	PRESIDENT		17-NOV-1981 00:00:00	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-1981 00:00:00	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-1987 00:00:00	1100		20
7900	JAMES	CLERK	7698	03-DEC-1981 00:00:00	950		30
7902	FORD	ANALYST	7566	03-DEC-1981 00:00:00	3000		20
7934	MILLER	CLERK	7782	23-JAN-1982 00:00:00	1300		10

```
SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  e1.sal > ALL (SELECT e2.sal
                    FROM   emp e2
                    WHERE  e2.deptno = 20);
```

EMPNO	SAL
7839	5000

```
SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  NOT EXISTS (SELECT e2.sal
                  FROM   emp e2
                  WHERE  e2.deptno = 20
                  AND    e1.sal <= e2.sal);
```

## UNION / UNION ALL

**UNION** operatörü verilen şart ifadesine uygun olarak, çift kayıtları göz ardı eder ve her iki kümedeki tüm kayıtları seçer. Buna karşılık verilen şart ifadesine uygun olarak çift kayıtlarda dahil olmak üzere tüm kayıtların seçilmesi isteniyorsa **UNION ALL** operatörü kullanılır.

```
SELECT employee_id, job_id
FROM employees
UNION
SELECT employee_id, job_id
FROM job_history;
```

Results: Script Output Explain Autotrace

Results:

	EMPLOYEE_ID	JOB_ID
1	100	AD_PRES
2	101	AC_ACCOUNT
3	101	AC_MGR
4	101	AD_VP
5	102	AD_VP
6	102	IT_PROG
7	103	IT_PROG

```
SELECT employee_id, job_id
FROM employees
UNION ALL
SELECT employee_id, job_id
FROM job_history;
```

Results: Script Output Explain Autotrace

Results:

	EMPLOYEE_ID	JOB_ID
7	109	FI_ACCOUNT
8	110	FI_ACCOUNT
9	111	FI_ACCOUNT
10	112	FI_ACCOUNT
11	113	FI_ACCOUNT
12	108	FI_MGR
13	203	HR_REP
14	103	IT_PROG

# UNION

Birden fazla tablodaki kayıtların birleştirilerek listelenmesini sağlar. Select ifadesinden sonraki sütun sayıları eşit olmalı. Aynı kayıtlar listelenmez (UNION ALL ile aynı kayıtların listelenmesi sağlanır)

```
SELECT ogrenciNo AS a1, adi AS a2, soyadi AS a3  
FROM Ogrenciler  
WHERE adi LIKE 'Ayd%'
```

UNION

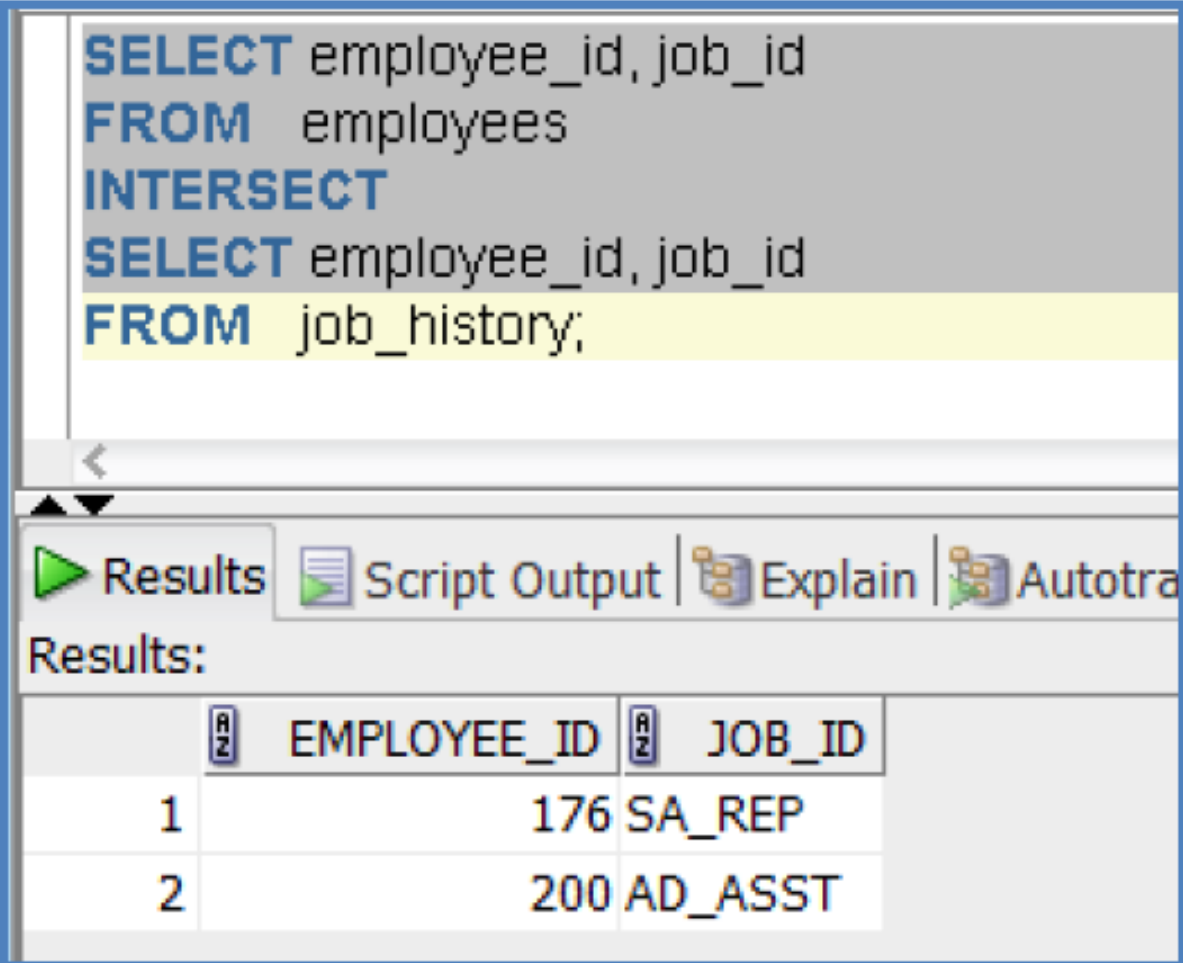
```
SELECT sicilNo, adi, soyadi  
FROM Personel;
```

a1	a2	a3
100000000003	Aydın	Mert
100000000007	Aydın	Mete
100000000008	Aydın	Aymaz
100000000011	Aydın	Aygün
100000000009	Aydınay	Aymaz
100000000010	Aydınay	Aygün
44556	Mertt	Korkmazt
2222	Burak	İnner
0	Mehmet	Yılmaz
0	Burak Alparslan	Yorulmaz
0	Ayşe	as
0	Melek	Şahin



### INTERSECT

Her iki SELECT sorgusundan dönen kayıtların eşit olanlarını bulmaya yarar.



The screenshot shows a SQL query editor with the following query:

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

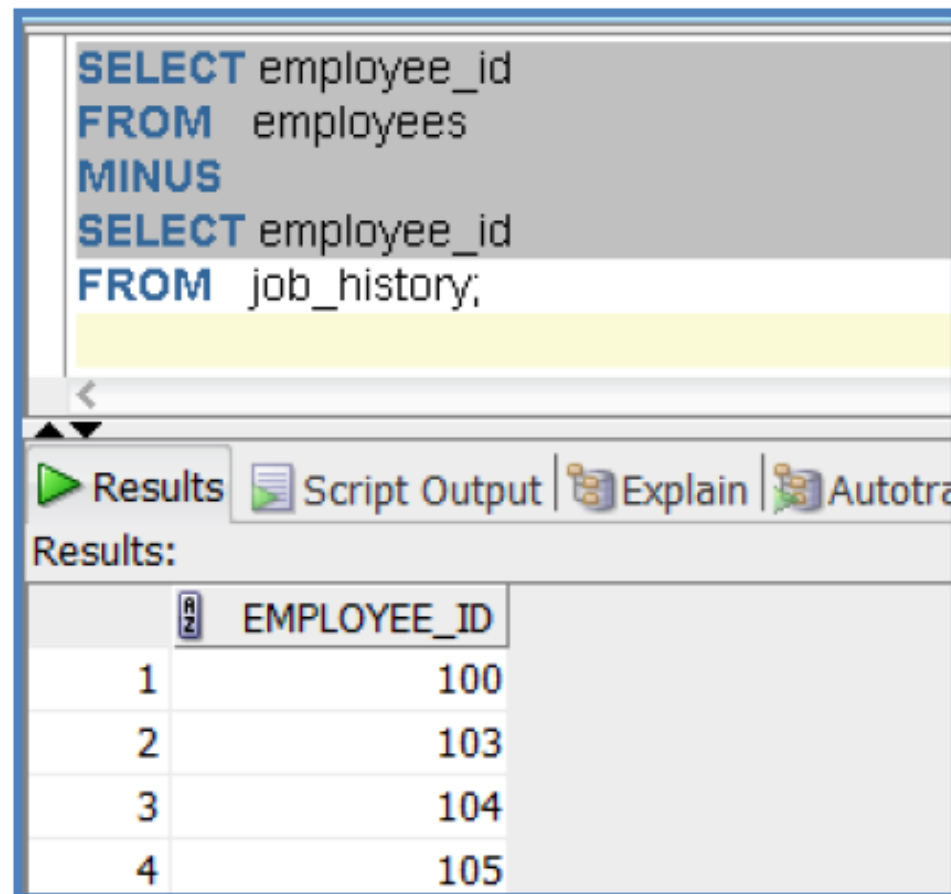
Below the query editor, there are tabs for "Results", "Script Output", "Explain", and "Autotrace". The "Results" tab is selected, showing the following results:

Results:

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

## Karmaşık Sorgular (MINUS =ORACLE, EXCEPT=MSSQL)

Birinci sorgudan elde edilen sonuçları ikinci sorgudan elde edilen sonuçlardan çıkarır.



The screenshot shows a SQL query editor window. The query text is:

```
SELECT employee_id
FROM employees
MINUS
SELECT employee_id
FROM job_history;
```

Below the query editor, there are tabs for 'Results', 'Script Output', 'Explain', and 'Autotrace'. The 'Results' tab is selected, and it displays the following results:

	EMPLOYEE_ID
1	100
2	103
3	104
4	105

## EXCEPT (MSSQL)



Tablo 1

Sütun 1
a
b
c
d

Tablo 2

Sütun1
a
b
c

□

Tablo 1 EXCEPT  
Tablo 2

**SORU:** Satış bölümündeki personel adlarından, mühendislik bölümünde bulunmayanları listeleyiniz:  
(Satış için bol\_no 1, mühendislik için bol\_no 2 olduğunu varsayalım.)

**ÇÖZÜM:**

```
SELECT * FROM  
    (SELECT ad FROM Personel  
     WHERE bol_no=1  
    EXCEPT  
    SELECT ad FROM Personel  
     WHERE bol_no=2);
```

# NESTED SELECT

Parça numarası 24 olan parçayı kullanan projelerde çalışan personeli listeleyiniz.

Proje

proje_ad	proj_no	yer	bl_no
1	1	İstanbul	4
2	2	İstanbul	4
3	3	Ankara	5
4	4	Ankara	5
5	5	İzmir	4

Parça

par_no	par_ad	pr_no	fiyat	ağırlık
24	Vida	2	2000	500
24	Vida	4	2000	500
37	Civata	2	6000	800
87	Conta	2	7000	5000
112	Pim	5	6000	70

Personel

sicil	sosy_g_no	ad	soyad	dog_tar	bol_no	adres
117	274251	Ali	Can	05/01/60	4	Akar sok. 2 Fatih
247	527241	Hasan	Okan	04/07/62	4	Merk cad. 3 Pendik
348	5276672	Ayşe	Pekcan	04/08/65	5	.....
548	443211	Akın	Pekol	07/02/70	4	.....
1148	52625	Mert	Caner	04/08/70	5	.....

Çalışma

per_s_g_no	proje_no	saat
274251	1	250
527241	2	350
527672	3	400
443211	5	300
527625	4	250

```
SELECT *  
FROM Personel  
WHERE sosy_g_no IN (SELECT Per_s_g_no  
                     FROM Parça, Çalışma  
                     WHERE pr_no=proj_no  
                     AND parça_no=24);
```

# NESTED SELECT

Fatih'te oturan personelin çalıştığı projelerin adları ve yerlerini listeleyiniz.

Proje

proje_ad	proj_no	yer	bl_no
1	1	İstanbul	4
2	2	İstanbul	4
3	3	Ankara	5
4	4	Ankara	5
5	5	İzmir	4

Parça

par_no	par_ad	pr_no	fiyat	ağırlık
24	Vida	2	2000	500
24	Vida	4	2000	500
37	Civata	2	6000	800
87	Conta	2	7000	5000
112	Pim	5	6000	70

Personel

sicil	sosy_g_no	ad	soyad	dog_tar	bol_no	adres
117	274251	Ali	Can	05/01/60	4	Akar sok. 2 Fatih
247	527241	Hasan	Okan	04/07/62	4	Merk cad. 3 Pendik
348	5276672	Ayşe	Pekcan	04/08/65	5	.....
548	443211	Akın	Pekol	07/02/70	4	.....
1148	52625	Mert	Caner	04/08/70	5	.....

Çalışma

per_s_g_no	proje_no	saat
274251	1	250
527241	2	350
527672	3	400
443211	5	300
527625	4	250

```
SELECT proje_ad,yer
FROM Proje
WHERE proj_no IN(SELECT proje_no
                  FROM Personel,Çalışma
                  WHERE sosy_g_no=Per_s_g_no
                  AND adres LIKE '%Fatih%');
```

# EXIST

OgrenciKayit tablosunda kaydı bulunan ogrencilerin listesi

```
SELECT *  
FROM Ogrenciler  
WHERE EXISTS (SELECT * FROM OgrenciKayit  
               WHERE ogrenciler.ogrenciNo = ogrenciKayit.ogrenciNo) ;
```

Burada herhangi bir şey olabilir.

OgrenciKayit tablosunda kaydı bulunmayan ogrencilerin listesi

```
SELECT *  
FROM Ogrenciler  
WHERE NOT EXISTS (SELECT * FROM OgrenciKayit  
                  WHERE ogrenciler.ogrenciNo = ogrenciKayit.ogrenciNo) ;
```

# VIEW OLUSTURMAK

- ✓ SELECT işlemi sonucunda oluşan kayıtları içeren sanal tabloya view denir.
- ✓ Dinamiktir. View ile oluşturulan tablo her çalıştığında view ı oluşturan ifadeler yeniden çalıştırılır.
- ✓ Karmaşık sorguları basit hale getirir.
- ✓ Güvenlik nedeniyle kullanılabilir. (Öğrenciler notları sadece görüntüleyebilsin, değiştiremesin v.s.)

# VIEW OLUSTURMAK

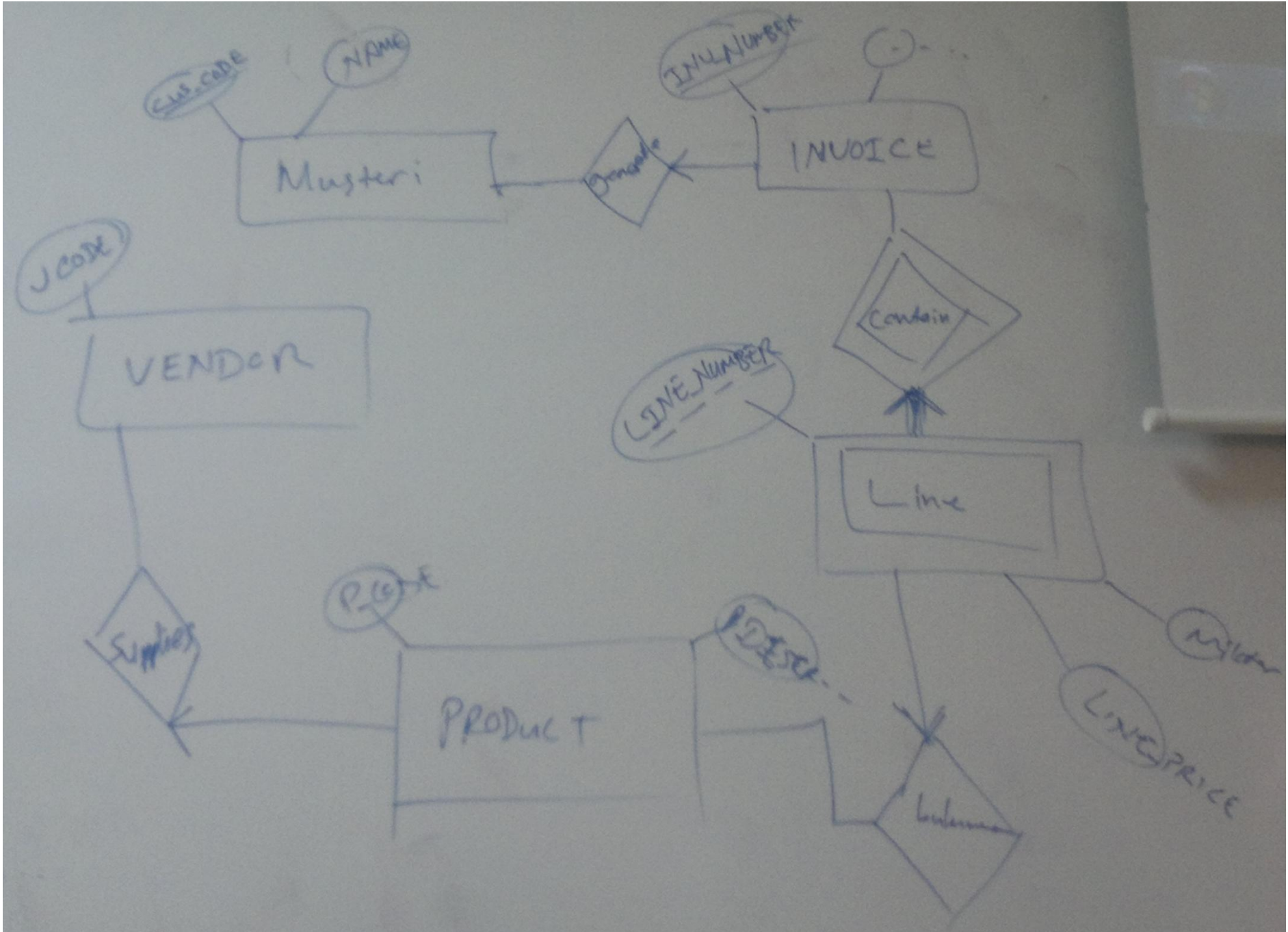
```
CREATE VIEW `dershaneyonetimsistemi`.`OgrenciGenelBilgiler` AS
SELECT
    ogrenciler.ogrenciNo
    , ogrenciler.adi
    , ogrenciler.soyadi
    , iller.ilAdi
    , ilceler.ilceAdi
    , ogrenimdurumu.ogrenimDurumu
FROM
    dershaneyonetimsistemi.ogrenciler
    INNER JOIN dershaneyonetimsistemi.iller
        ON (ogrenciler.il = iller.ilKodu)
    INNER JOIN dershaneyonetimsistemi.ilceler
        ON (ilceler.il = iller.ilKodu) AND (ogrenciler.ilce = ilceler.ilceKodu)
    INNER JOIN dershaneyonetimsistemi.ogrenimdurumu
        ON (ogrenciler.ogrenimDurumu = ogrenimdurumu.ogrenimNo);
```

1 Messages 2 Table Data 3 Info 4 History					
All Rows Rows in a Range First Row: 0 No. of Rows: 50 Refresh					
ogrenciNo	adi	soyadi	ilAdi	ilceAdi	ogrenimDurumu
00000000002	ahmet	metin	Bilinmiyor	Bilinmiyor	Girilmemiş
00000000061	Ahmet	Mete	Bilinmiyor	Bilinmiyor	Girilmemiş
10000000001	Ayla	Mert	Bilinmiyor	Bilinmiyor	Girilmemiş
10000000002	Aylin	Mert	Bilinmiyor	Bilinmiyor	Girilmemiş
10000000003	Aydın	Mert	Bilinmiyor	Bilinmiyor	Girilmemiş
10000000004	Ayhan	Mert	Bilinmiyor	Bilinmiyor	Girilmemiş
10000000005	Ayhan	Metin	Bilinmiyor	Bilinmiyor	Girilmemiş
10000000006	Ayhan	Mete	Bilinmiyor	Bilinmiyor	Girilmemiş
10000000007	Ayhan	Mete	Bilinmiyor	Bilinmiyor	Girilmemiş

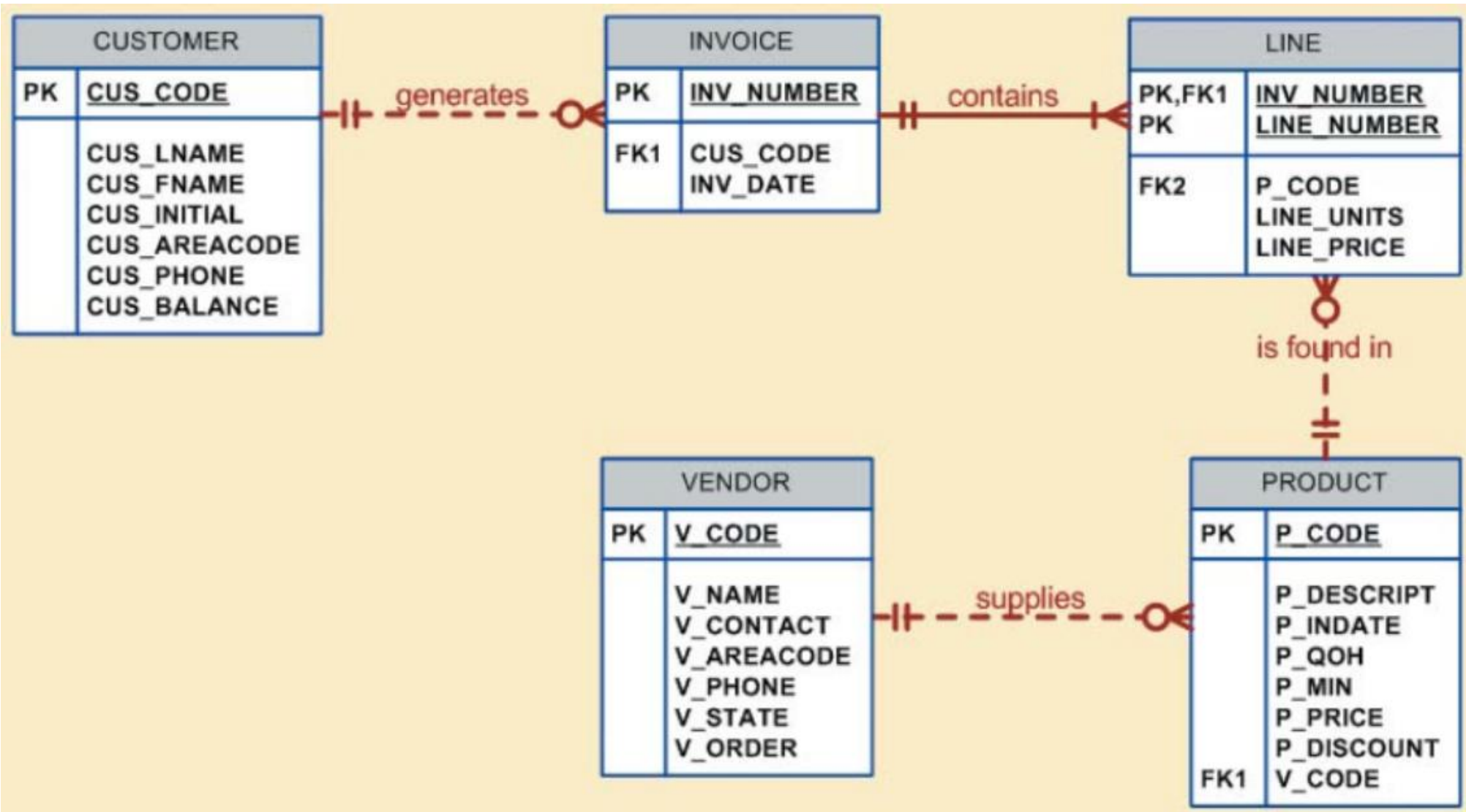
```
SELECT *
FROM
OgrenciGenelBilgiler
;
```



# Ürün Sipariş Sistemi



# Ürün Sipariş Sistemi



# DML İle Alt Sorgu Kullanımı

```
INSERT INTO PRODUCT  
  SELECT * FROM P;
```

```
UPDATE PRODUCT  
SET    P_PRICE = (SELECT AVG(P_PRICE)  
                  FROM PRODUCT)  
WHERE V_CODE IN (SELECT V_CODE  
                  FROM VENDOR  
                  WHERE V_AREACODE = '615')
```

```
DELETE FROM PRODUCT  
WHERE V_CODE IN (SELECT V_CODE  
                  FROM VENDOR  
                  WHERE V_AREACODE = '615')
```

## Where İle Alt Sorgu (Tek Değer Döndüren) Kullanımı

```
SELECT    P_CODE, P_PRICE FROM PRODUCT
WHERE     P_PRICE >= (SELECT AVG(P_PRICE) FROM PRODUCT);
```

Tek olarak =,<,> v.s. gibi ifadeler kullanılıyor ise alt sorgular sonucunda tek alan ve tek satır dönmeli ve tipi uygun olmalı. Aksi halde hata verir.

```
SELECT    DISTINCT CUS_CODE, CUS_LNAME, CUS_FNAME
FROM      CUSTOMER JOIN INVOICE USING (CUS_CODE)
           JOIN LINE USING (INV_NUMBER)
           JOIN PRODUCT USING (P_CODE)
WHERE     P_CODE = (SELECT P_CODE FROM PRODUCT WHERE P_DESCRIPT = 'Claw hammer');
```

## Where İle Alt Sorgu (Çok Değer Döndüren) Kullanımı

```
SELECT    DISTINCT CUS_CODE, CUS_LNAME, CUS_FNAME
FROM      CUSTOMER JOIN INVOICE USING (CUS_CODE)
           JOIN LINE USING (INV_NUMBER)
           JOIN PRODUCT USING (P_CODE)
WHERE     P_CODE IN ( SELECT    P_CODE FROM PRODUCT
                       WHERE     P_DESCRIPT LIKE '%hammer%'
                       OR        P_DESCRIPT LIKE '%saw%');
```

```
SELECT    P_CODE, P_QOH * P_PRICE
FROM      PRODUCT
WHERE     P_QOH * P_PRICE > ALL (SELECT P_QOH * P_PRICE
                                FROM PRODUCT
                                WHERE V_CODE IN (SELECT V_CODE
                                                  FROM VENDOR
                                                  WHERE V_STATE = 'FL'));
```

## Having ile Alt Sorgu Kullanımı

```
SELECT      P_CODE, SUM(LINE_UNITS)
FROM        LINE
GROUP BY    P_CODE
HAVING      SUM(LINE_UNITS) > (SELECT AVG(LINE_UNITS) FROM LINE);
```



## From İle Alt Sorgu Kullanımı

```
SELECT      DISTINCT CUSTOMER.CUS_CODE, CUSTOMER.CUS_LNAME
FROM        CUSTOMER,
            (SELECT INVOICE.CUS_CODE FROM INVOICE NATURAL JOIN LINE
             WHERE P_CODE = '13-Q2/P2') CP1,
            (SELECT INVOICE.CUS_CODE FROM INVOICE NATURAL JOIN LINE
             WHERE P_CODE = '23109-HB') CP2
WHERE       CUSTOMER.CUS_CODE = CP1.CUS_CODE AND CP1.CUS_CODE = CP2.CUS_CODE;
```

```
CREATE VIEW CP1 AS
```

```
    SELECT      INVOICE.CUS_CODE FROM INVOICE NATURAL JOIN LINE
    WHERE       P_CODE = '13-Q2/P2';
```

```
CREATE VIEW CP2 AS
```

```
    SELECT      INVOICE.CUS_CODE FROM INVOICE NATURAL JOIN LINE
    WHERE       P_CODE = '23109-HB';
```

```
SELECT      DISTINCT CUS_CODE, CUS_LNAME
FROM        CUSTOMER NATURAL JOIN CP1 NATURAL JOIN CP2;
```

## Inline Alt Sorgu Kullanımı

Alt sorgular sonucunda tek alan ve tek satır dönmeli. Aksi halde hata verir. DIFF hesaplanırken AVGPRICE kısa adı kullanılmamalı.

```
SELECT      P_CODE, P_PRICE, (SELECT AVG(P_PRICE) FROM PRODUCT) AS AVGPRICE,  
            P_PRICE - (SELECT AVG(P_PRICE) FROM PRODUCT) AS DIFF  
FROM        PRODUCT;
```

```
SELECT      P_CODE, SUM(LINE_UNITS * LINE_PRICE) AS SALES,  
            (SELECT COUNT(*) FROM EMPLOYEE) AS ECOUNT,  
            SUM(LINE_UNITS * LINE_PRICE)/(SELECT COUNT(*) FROM EMPLOYEE) AS CONTRIB  
FROM        LINE  
GROUP BY    P_CODE;
```

```
SELECT      P_CODE, SALES, ECOUNT, SALES/ECOUNT AS CONTRIB  
FROM        (SELECT P_CODE, SUM(LINE_UNITS * LINE_PRICE) AS SALES,  
                (SELECT COUNT(*) FROM EMPLOYEE) AS ECOUNT  
              FROM      LINE  
              GROUP BY P_CODE);
```



## İlintili (Correlated) Alt Sorgu Kullanımı

İç içe döngülerdeki gibi dış sorgunun her bir satırı iç sorguya gönderilerek iç sorgunun çalıştırılması sağlanır.

Aşağıdaki örnekte; dış sorgunun her satırı için iç sorgu çalışır ve dış sorgudaki P\_CODE değeri için ortalama miktarı bulur. Bu değer LINE\_UNITS değerinden daha küçük ise gösterilmek üzere sonuç kümesine (result set) eklenir.

```
SELECT      INV_NUMBER, P_CODE, LINE_UNITS
FROM        LINE LS
WHERE       LS.LINE_UNITS > (SELECT      AVG(LINE_UNITS)
                              FROM        LINE LA
                              WHERE       LA.P_CODE = LS.P_CODE);
```

```
SELECT      CUS_CODE, CUS_LNAME, CUS_FNAME
FROM        CUSTOMER
WHERE       EXISTS      (SELECT      CUS_CODE FROM INVOICE
                        WHERE       INVOICE.CUS_CODE = CUSTOMER.CUS_CODE);
```

# Kaynaklar

- ✓ Carlos Coronel, Steven Morris, and Peter Rob, Database Systems: Design, Implementation, and Management, Cengage Learning.
- ✓ Ümit Kocabıçak, Ders Notları, Sakarya Üniversitesi Bilgisayar ve Bilişim Bilimleri Fakültesi Bilgisayar Mühendisliği Bölümü.
- ✓ Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, Mc Graw Hill