

# **Ondokuz Mayıs Üniversitesi**

**Bilgisayar Mühendisliği**



**Aleyna KAHRAMAN**

**20060355**

**Deney Föyü-3**

**Hami SATILMIŞ**

---

## Ö Z E T

---

Bu rapor, bir veritabanı oluşturma ve veri sorgulama sürecini kapsamaktadır. İlk olarak, veritabanı yönetim sistemi üzerinde SQL kodları kullanılarak bir dizi tablo oluşturulmuştur. Oluşturulan tablolar arasında ilişkiler kurulmuş ve veri bütünlüğünü sağlamak amacıyla FOREIGN KEY kısıtlamaları tanımlanmıştır.

Daha sonra, oluşturulan tablolara örnek veriler eklenmiş ve tablolar arasındaki ilişkiler doğrulanmıştır. Eklenen veriler, çalışanların ad, soyad, maaş, birim ve unvan bilgilerini içermektedir.

Son olarak, istenen analizler için SQL sorguları kullanılarak veri sorgulanmıştır. Bu analizler arasında en yüksek maaşı alan çalışanların belirlenmesi, belirli bir maaş aralığındaki çalışanların listelenmesi ve belirli unvanlara sahip çalışanların bilgilerinin görüntülenmesi bulunmaktadır.

---

## İÇİNDEKİLER

---

### I GİRİŞ

1 Amaç	2
--------	---

### II MATERYAL

2 Uygulamalar	4
2.1 Microsoft SQL Server 2017 . . . . .	4
2.2 SQL Server Management Studio (SSMS) . . . . .	4
3 İSTENİLENLER	6
3.1 Soru-1 . . . . .	6
3.1.1 Cevap-1 . . . . .	6
3.2 Soru-2 . . . . .	8
3.2.1 Cevap-2 . . . . .	8
3.3 Soru-3 . . . . .	10
3.3.1 Cevap-3 . . . . .	10
3.4 Soru-4 . . . . .	11
3.4.1 Cevap-4 . . . . .	11
3.5 Soru-5 . . . . .	12
3.5.1 Cevap-5 . . . . .	12
3.6 Soru-6 . . . . .	13
3.6.1 Cevap-6 . . . . .	13

3.7 Soru-7 . . . . .	14
3.7.1 Cevap-7 . . . . .	14
3.8 Soru-8 . . . . .	15
3.8.1 Cevap-8 . . . . .	15
3.9 Soru-9 . . . . .	16
3.9.1 Cevap-9 . . . . .	16
3.10 Soru-10 . . . . .	17
3.10.1 Cevap-10 . . . . .	17
 <b>III SONUÇ</b>	
4 Sonuç	20
KAYNAKÇA	21

---

## ŞEKİLLER LİSTESİ

---

1	Veritabanı UML . . . . .	7
2	Tablolar . . . . .	7
3	Birim Ekleme . . . . .	8
4	Çalışan Ekleme . . . . .	9
5	Unvan Ekleme . . . . .	9
6	İkramiye Ekleme . . . . .	9
7	Cevap-3 . . . . .	10
8	Cevap-4 . . . . .	11
9	Cevap-5 . . . . .	12
10	Cevap-6 . . . . .	13
11	Cevap-7 . . . . .	14
12	Cevap-8 . . . . .	16
13	Cevap-9 . . . . .	17
14	Cevap-10 . . . . .	18

## BÖLÜM: İ

### GİRİŞ

---

## A M A Ç

---

Bu föy, veritabanı yönetimi ve SQL sorgulama konularında öğrencilerin temel becerilerini geliştirmeyi hedeflemektedir. Bu süreçte, öğrencilerin veritabanı oluşturma, ilişkiler kurma, veri ekleyip sorgulama, veri analizi yapma ve veri odaklı karar alma gibi temel konularda pratik yapmaları amaçlanmaktadır. Föyün amacı, öğrencilerin bu süreçte edindikleri bilgi ve becerilerle veri tabanı yönetimi alanında kendilerini geliştirmelerini sağlamaktır.

BÖLÜM: İİ

MATERYAL



---

## UYGULAMALAR

---

Microsoft SQL Server 2017 Kurulumu ve SQL Server Management Studio Yüklemesi yapılmıştır.

### 2.1 *Microsoft SQL Server 2017*

Microsoft SQL Server 2017, işletmelerin veri yönetimi, analizi, raporlama ve iş zekası gibi çeşitli gereksinimlerini karşılayan bir ilişkisel veritabanı yönetim sistemidir. Büyük miktarlarda veri depolama, veri güvenliği sağlama, veri analizi yapma, iş zekası çözümleri oluşturma ve uygulama geliştirme için kullanılır. Ayrıca bulut tabanlı platformlarla entegrasyon sağlayarak ölçeklenebilirlik ve esneklik sunar..

### 2.2 *SQL Server Management Studio (SSMS)*

SQL Server Management Studio (SSMS), Microsoft SQL Server'ı yönetmek, veritabanı nesnelerini yönetmek, sorguları çalıştırmak, veri işlemek ve performansı izlemek için kullanılan

entegre bir geliştirme ortamıdır. Veritabanı yöneticileri ve geliştiriciler için temel bir araç olan SSMS, SQL Server'a erişimi kolaylaştırır ve yönetimi daha verimli hale getirir.

---

## İSTENİLENLER

---

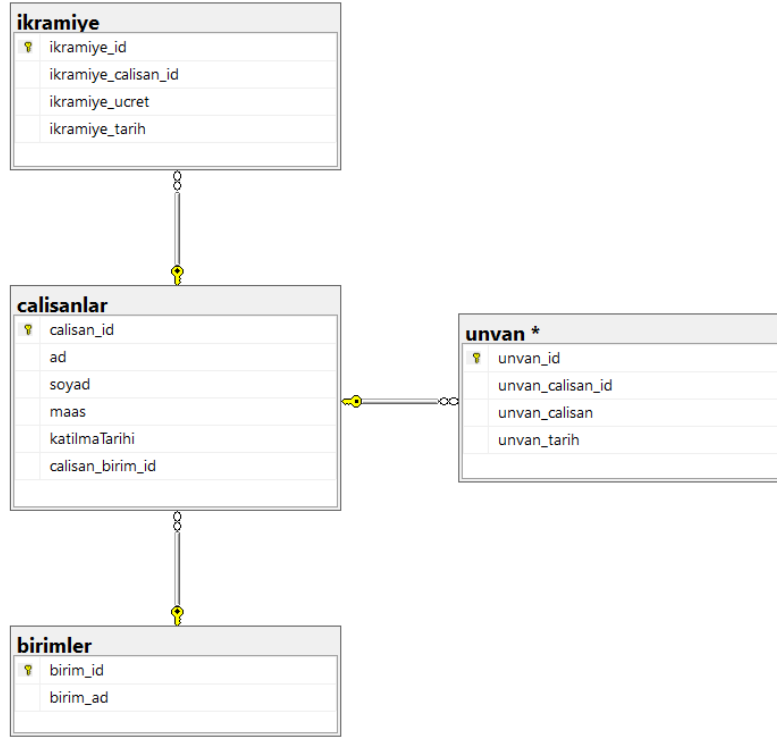
Kodların bulunduğu github linki: [\[1\]](#)

### 3.1 Soru-1

Verilen diyagramın veritabanını SQL Server ortamında oluşturunuz. Veritabanını oluştururken tablolar arasındaki ilişkilere dikkat ediniz, özniteliklerin veri tiplerini doğru tanımlayınız, tablolarda bulunan Primary Key (birim\_id ve calisan\_id) ve Foreign Key (calisan\_birim\_id, unvan\_calisan\_id ve ikramiye\_calisan\_id) özniteliklerinin tablolar arasındaki ilişkileri belirtiniz.

#### 3.1.1 Cevap-1

Verilen kodlarla oluşturulan tablolar, şekil[2]'de gösterilen diyagramda bulunmaktadır. Elde edilen UML diyagramı, şekil[1]'de gösterilmiştir. Bu süreçte, birimler, çalışanlar, ikramiyeler ve unvanlar gibi tablolar tanımlanmış ve bu tablolar arasında uygun ilişkiler kurulmuştur.



Şekil 1: Veritabanı UML

```

SQLQuery3.sql - DE...RESS.foy3 (sa (54))*  DESKTOP-VMO3C7M\...y3 - dbo.Ta
-- Create Table birimler (
    birim_id INT PRIMARY KEY,
    birim_ad CHAR(25) NULL
);

-- Calisanlar tablosu
-- Create Table calisanlar (
    calisan_id INT PRIMARY KEY,
    ad CHAR(25) NULL,
    soyad CHAR(25) NULL,
    maas INT NULL,
    katilmaTarihi DATETIME NULL,
    calisan_birim_id INT NULL,
    FOREIGN KEY (calisan_birim_id) REFERENCES birimler(birim_id)
);

-- Ikramiye tablosu
-- Create Table ikramiye (
    ikramiye_id INT PRIMARY KEY IDENTITY,
    ikramiye_calisan_id INT NULL,
    ikramiye_ucret INT NULL,
    ikramiye_tarih DATETIME NULL,
    FOREIGN KEY (ikramiye_calisan_id) REFERENCES calisanlar(calisan_id)
);

-- Unvan tablosu
-- Create Table unvan (
    unvan_id INT PRIMARY KEY IDENTITY,
    unvan_calisan_id INT NULL,
    unvan_calisan CHAR(25) NULL,
    unvan_tarih DATETIME NULL,
    FOREIGN KEY (unvan_calisan_id) REFERENCES calisanlar(calisan_id)
);

```

70 %

Messages

Commands completed successfully.

Completion time: 2024-10-31T02:29:54.4456211+03:00

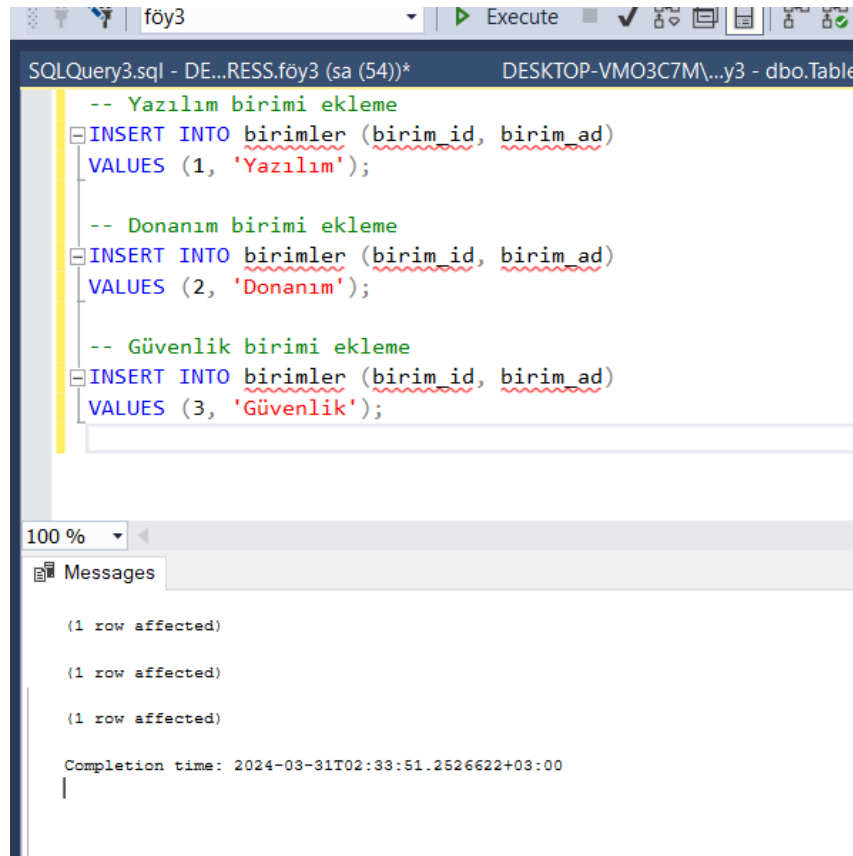
Şekil 2: Tablolar

### 3.2 Soru-2

Veritabanına veri eklemek

#### 3.2.1 Cevap-2

Eklenmesi istenen veriler [şekil\[3,5,4, 6\]](#)'de gösterildiği şekilde eklenmiştir.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query in the 'SQLQuery3.sql' file, which is connected to the 'föy3' database. The query consists of three INSERT statements to add data to the 'birimler' table. The bottom pane shows the execution results, indicating that each statement successfully affected one row. The completion time is 2024-03-31T02:33:51.2526622+03:00.

```
-- Yazılım birimi ekleme
INSERT INTO birimler (birim_id, birim_ad)
VALUES (1, 'Yazılım');

-- Donanım birimi ekleme
INSERT INTO birimler (birim_id, birim_ad)
VALUES (2, 'Donanım');

-- Güvenlik birimi ekleme
INSERT INTO birimler (birim_id, birim_ad)
VALUES (3, 'Güvenlik');
```

100 %

Messages

(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2024-03-31T02:33:51.2526622+03:00

Şekil 3: Birim Ekleme

```

SQLQuery4.sql - DE...RESS.foy3 (sa (65))* x SQLQuery3.sql - DE...RESS.foy3 (sa (54))*
INSERT INTO calisanlar (calisan_id, ad, soyad, maas, katilmaTarihi, calisan_birim_id) VALUES
(1, 'İsmail', 'İşeri', 100000, '2014-02-20', 1),
(2, 'Hami', 'Satılmış', 80000, '2014-06-11', 1),
(3, 'Durmus', 'Şahin', 300000, '2023-02-20', 2),
(4, 'Kağan', 'Yazan', 500000, '2014-02-20', 3),
(5, 'Meryem', 'Soysaldı', 500000, '2014-06-11', 3),
(6, 'Duygu', 'Akşehin', 200000, '2014-06-11', 2),
(7, 'Kübra', 'Seyhan', 75000, '2014-01-20', 1),
(8, 'Gülcan', 'Yıldız', 90000, '2014-04-11', 3);
S
100 %
Messages
(8 rows affected)
Completion time: 2024-03-31T02:50:08.4982632+03:00

```

Şekil 4: Çalışan Ekleme

```

SQLQuery4.sql - DE...RESS.foy3 (sa (65))* x SQLQuery3.sql - DE...RESS.foy3 (sa (54))*
-- 8 unvan ekleyelim
INSERT INTO unvan (unvan_calisan_id, unvan_calisan, unvan_tarih) VALUES
(1, 'Yönetici', '2016-02-20'),
(2, 'Personel', '2016-06-11'),
(8, 'Personel', '2016-06-11'),
(5, 'Müdür', '2016-06-11'),
(4, 'Yönetici Yardımcısı', '2016-06-11'),
(7, 'Personel', '2016-06-11'),
(6, 'Takım Lideri', '2016-06-11'),
(3, 'Takım Lideri', '2016-06-11');
100 %
Messages
(8 rows affected)
Completion time: 2024-03-31T03:01:23.4617106+03:00

```

Şekil 5: Unvan Ekleme

```

föy3 Execute
SQLQuery4.sql - DE...RESS.foy3 (sa (65))* x SQLQuery3.sql - DE...RESS.foy3 (sa (54))*
-- 5 ikramiye verisi ekleme
INSERT INTO ikramiye (ikramiye_calisan_id, ikramiye_ucret, ikramiye_tarih) VALUES
(1, 5000, '2016-02-20'),
(2, 3000, '2016-06-11'),
(3, 4000, '2016-02-20'),
(1, 4500, '2016-02-20'),
(2, 3500, '2016-06-11');
100 %
Messages
(5 rows affected)
Completion time: 2024-03-31T02:57:16.5113052+03:00

```

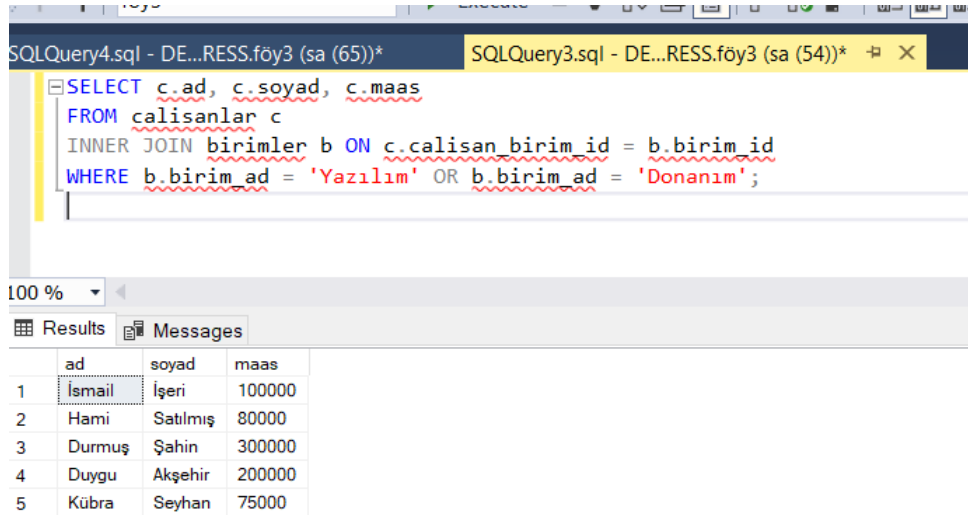
Şekil 6: İkramiye Ekleme

### 3.3 Soru-3

“Yazılım” veya “Donanım” birimlerinde çalışanların ad, soyad ve maaş bilgilerini listeleyen SQL sorgusu

#### 3.3.1 Cevap-3

Bu SQL sorgusu şekil[7]’te, "calisanlar" tablosundan çalışanların adını (ad), soyadını (soyad) ve maaşını (maas) seçer. İç birleştirme (INNER JOIN) kullanarak, "calisanlar" tablosundaki "calisan\_birim\_id" sütununu "birimler" tablosundaki "birim\_id" sütunuyla eşler. Bu eşleşme, "birim\_ad" sütununda 'Yazılım' veya 'Donanım' olan birimlerle sınırlanmıştır. Sonuçlar, çalışanların adı, soyadı ve maaşıyla birlikte, bu şartı sağlayan birimlerle ilişkilendirilir.



The screenshot shows a SQL query editor with two tabs. The active tab is 'SQLQuery4.sql - DE...RESS.föy3 (sa (65))\*'. The query is as follows:

```
SELECT c.ad, c.soyad, c.maas
FROM calisanlar c
INNER JOIN birimler b ON c.calisan_birim_id = b.birim_id
WHERE b.birim_ad = 'Yazılım' OR b.birim_ad = 'Donanım';
```

Below the query editor, the 'Results' tab is selected, displaying the following data:

	ad	soyad	maas
1	İsmail	İşeri	100000
2	Hami	Satılmış	80000
3	Durmuş	Şahin	300000
4	Duygu	Akşehir	200000
5	Kübra	Seyhan	75000

Şekil 7: Cevap-3

### 3.4 Soru-4

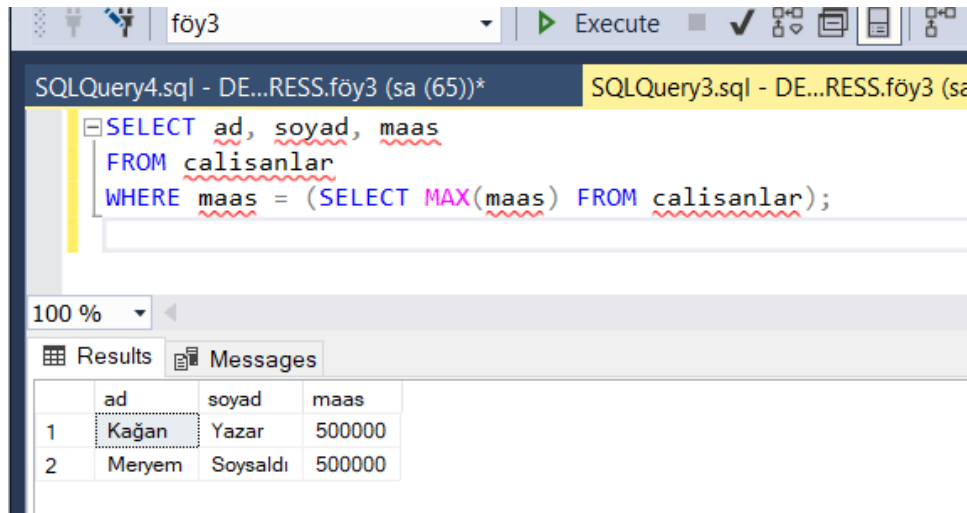
Maaşı en yüksek olan çalışanların ad, soyad ve maaş bilgilerini listeleyen SQL sorgusu

#### 3.4.1 Cevap-4

Bu SQL sorgusu şekil[8]'te, "calisanlar" tablosundan çalışanların adını (ad), soyadını (soyad) ve maaşını (maas) seçer. Ancak, WHERE koşulu içinde bir alt sorgu kullanılmıştır.

Alt sorgu şunu yapar: "calisanlar" tablosundaki en yüksek maaşı bulur. Yani, MAX() fonksiyonu kullanılarak tüm maaşlar arasından en yüksek olanı belirler.

Ana sorgu, bu en yüksek maaşa eşit olan çalışanları seçer. Yani, "calisanlar" tablosundaki maaşı en yüksek olan çalışanların adını, soyadını ve maaşını döndürür.



The screenshot shows a SQL IDE window with a query editor and a results pane. The query editor contains the following SQL code:

```
SELECT ad, soyad, maas
FROM calisanlar
WHERE maas = (SELECT MAX(maas) FROM calisanlar);
```

The results pane shows the following data:

	ad	soyad	maas
1	Kağan	Yazar	500000
2	Meryem	Soysaldı	500000

Şekil 8: Cevap-4



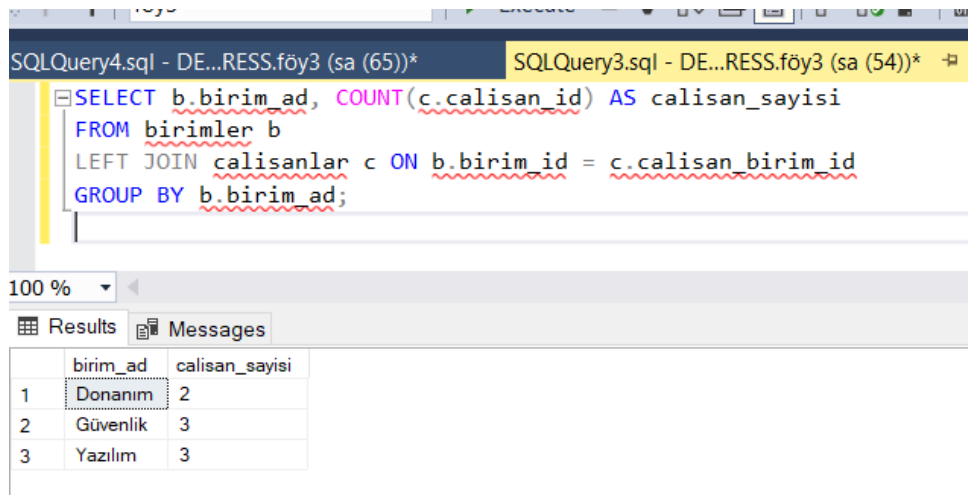
### 3.5 Soru-5

Birimlerin her birinde kaç adet çalışan olduğunu ve birimlerin isimlerini listeleyen sorgu

#### 3.5.1 Cevap-5

Bu SQL sorgusu şekil[9]'te, "birimler" tablosundan birim adını (birim\_ad) ve "calisanlar" tablosundan çalışanların sayısını hesaplar. LEFT JOIN kullanarak, "birimler" tablosundaki birimlerle "calisanlar" tablosundaki çalışanları birleştirir. Burada, LEFT JOIN kullanıldığı için, birimler tablosunda eşleşme olmasa bile (yani, çalışan olmayan birimler de dahil) tüm birimleri döndürür.

GROUP BY kullanarak, birim adlarına göre gruplandırır. Her bir grup için, birim adı ve o birime bağlı çalışanların sayısı (calisan\_sayisi) hesaplanır ve döndürülür. Bu sayede her bir birim için çalışan sayısı bulunmuş olur.



The screenshot shows a SQL query editor with two tabs: 'SQLQuery4.sql' and 'SQLQuery3.sql'. The active tab is 'SQLQuery4.sql', which contains the following SQL query:

```
SELECT b.birim_ad, COUNT(c.calisan_id) AS calisan_sayisi
FROM birimler b
LEFT JOIN calisanlar c ON b.birim_id = c.calisan_birim_id
GROUP BY b.birim_ad;
```

Below the query editor, the 'Results' tab is selected, displaying the following table:

	birim_ad	calisan_sayisi
1	Donanim	2
2	Güvenlik	3
3	Yazılım	3

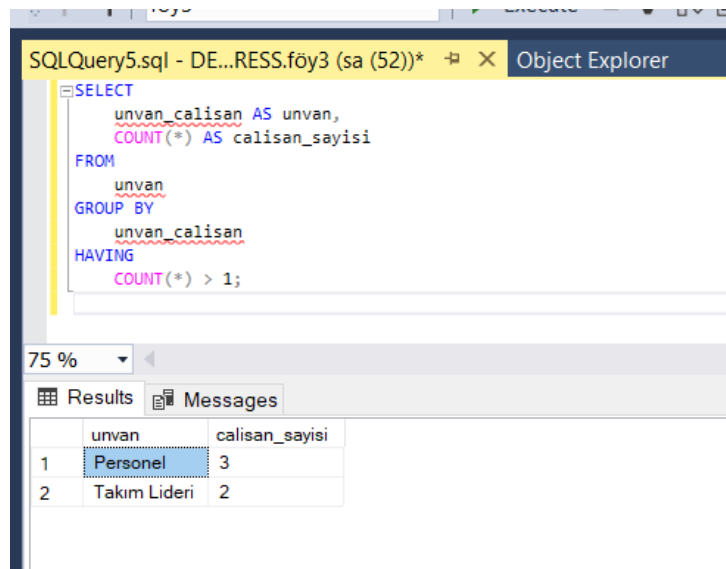
Şekil 9: Cevap-5

### 3.6 Soru-6

Birden fazla çalışana ait olan unvanların isimlerini ve o unvan altında çalışanların sayısını listeleyen sorgu

#### 3.6.1 Cevap-6

Bu SQL sorgusu şekil[10]'da, "unvan" tablosundan çalışan unvanlarını (unvan\_calisan) ve her unvan için kaç çalışanın olduğunu sayar. GROUP BY ifadesi unvanlara göre gruplandırma yapar. Her bir grup, bir unvana sahip çalışanları içerir. HAVING ifadesi ise GROUP BY'dan sonra gelen bir koşul ifadesidir. Bu koşul, gruplanmış sonuçlar üzerinde bir filtreleme işlemi yapar. Burada, COUNT(\*) > 1 ifadesi kullanılmıştır, yani her unvan için en az bir çalışanın olması gerektiği anlamına gelir. Bu şekilde, yalnızca birden fazla çalışana sahip unvanlar alınır. Sonuç olarak, sorgu, en az bir çalışana sahip olan unvanları ve bu unvanlara sahip çalışan sayısını döndürür.



The screenshot shows a SQL query window titled 'SQLQuery5.sql - DE...RESS.föy3 (sa (52))\*'. The query is as follows:

```
SELECT
    unvan_calisan AS unvan,
    COUNT(*) AS calisan_sayisi
FROM
    unvan
GROUP BY
    unvan_calisan
HAVING
    COUNT(*) > 1;
```

Below the query, the 'Results' tab is active, displaying the following data:

	unvan	calisan_sayisi
1	Personel	3
2	Takım Lideri	2

Şekil 10: Cevap-6

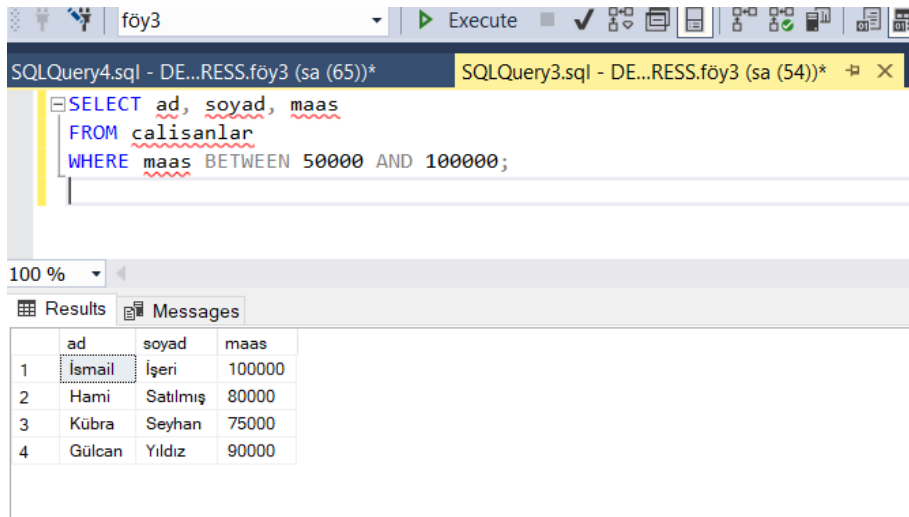
### 3.7 Soru-7

Maaşları “50000” ve “100000” arasında değişen çalışanların ad, soyad ve maaş bilgilerini listeleyen sorgu

#### 3.7.1 Cevap-7

Bu SQL sorgusu şekil[11]’de, "calisanlar" tablosundan çalışanların adını (ad), soyadını (soyad) ve maaşını (maas) seçer. WHERE koşulu, çalışan maaşlarının 50000 ile 100000 arasında (dahil) olmasını gerektirir.

Bu koşul, çalışanların maaşlarını filtrelemek için kullanılır. Yani, 50000 ile 100000 arasında maaş alan tüm çalışanları döndürür.



Şekil 11: Cevap-7

### 3.8 Soru-8

İkramiye hakkına sahip çalışanlara ait ad, soyad, birim, unvan ve ikramiye ücreti bilgilerini listeleyen sorgu

#### 3.8.1 Cevap-8

Bu SQL sorgusu şekil[12]'de, "calisanlar" tablosundan çalışanların adını (ad) ve soyadını (soyad), ayrıca bu çalışanların birimini (birim), unvanını (unvan) ve ikramiye ücretini (ikramiye\_ucret) seçer.

İlk INNER JOIN ifadesi, "calisanlar" tablosunu "ikramiye" tablosuyla birleştirir. İki tablo arasında "calisan\_id" sütununu eşleştirir ve çalışanların ikramiye bilgilerini alır.

İkinci INNER JOIN ifadesi, "calisanlar" tablosunu "birimler" tablosuyla birleştirir. Bu birleştirme işlemi "calisan\_birim\_id" ve "birim\_id" sütunlarını eşleştirerek çalışanların birim bilgilerini alır.

Üçüncü INNER JOIN ifadesi, "calisanlar" tablosunu "unvan" tablosuyla birleştirir. Bu birleştirme işlemi "calisan\_id" ve "unvan\_calisan\_id" sütunlarını eşleştirerek çalışanların unvan bilgilerini alır.

Sonuç olarak, sorgu, çalışanların adını, soyadını, birimini, unvanını ve ikramiye ücretini döndürür. Bu bilgiler, ilgili tablolardan alınarak birleştirilir ve istenen veriler elde edilir.

SQLQuery4.sql - DE...RESS.föy3 (sa (65))\* SQLQuery3.sql - DE...RESS.föy3 (sa (54))\*

```

SELECT c.ad, c.soyad, b.birim_ad AS birim, u.unvan_calisan AS unvan, i.ikramiye_ucret
FROM calisanlar c
INNER JOIN ikramiye i ON c.calisan_id = i.ikramiye_calisan_id
INNER JOIN birimler b ON c.calisan_birim_id = b.birim_id
INNER JOIN unvan u ON c.calisan_id = u.unvan_calisan_id;

```

100 %

Results Messages

	ad	soyad	birim	unvan	ikramiye_ucret
1	İsmail	İşeri	Yazılım	Yönetici	5000
2	İsmail	İşeri	Yazılım	Yönetici	4500
3	Hami	Satılmış	Yazılım	Personel	3000
4	Hami	Satılmış	Yazılım	Personel	3500
5	Durmuş	Şahin	Donanım	Takım Lideri	4000

Şekil 12: Cevap-8

### 3.9 Soru-9

Ünvanı “Yönetici” ve “Müdür” olan çalışanların ad, soyad ve ünvanlarını listeleyen sorgu

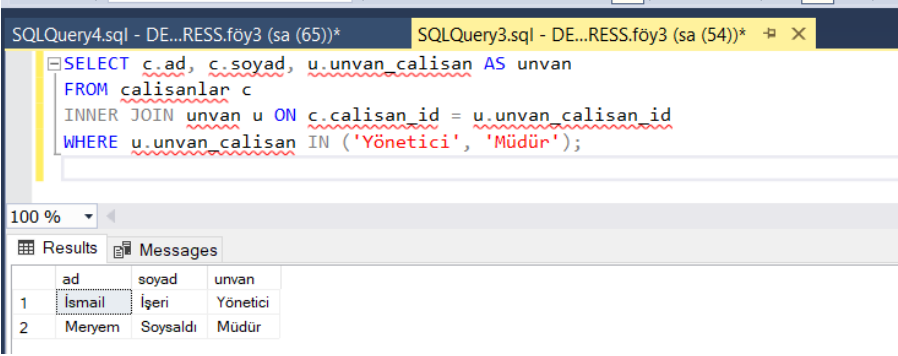
#### 3.9.1 Cevap-9

Bu SQL sorgusu şekil[13]’da, "calisanlar" tablosundan çalışanların adını (ad) ve soyadını (soyad), ayrıca bu çalışanların ünvanını (unvan\_calisan) seçer.

INNER JOIN ifadesi, "calisanlar" tablosunu "unvan" tablosuyla birleştirir. Bu birleştirme işlemi, "calisan\_id" sütununu "unvan\_calisan\_id" sütunuyla eşleştirerek çalışanların ünvan bilgilerini alır.

WHERE koşulu, çalışanların ünvanının 'Yönetici' veya 'Müdür' olduğu durumları filtreler.

Sonuç olarak, sorgu, 'Yönetici' veya 'Müdür' ünvanına sahip olan çalışanların adını, soyadını ve ünvanını döndürür.



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

```
SELECT c.ad, c.soyad, u.unvan_calisan AS unvan
FROM calisanlar c
INNER JOIN unvan u ON c.calisan_id = u.unvan_calisan_id
WHERE u.unvan_calisan IN ('Yönetici', 'Müdür');
```

The results pane shows the following data:

	ad	soyad	unvan
1	Ismail	İşeri	Yönetici
2	Meryem	Soysaldı	Müdür

Şekil 13: Cevap-9

### 3.10 Soru-10

Her bir birimde en yüksek maaş alan çalışanların ad, soyad ve maaş bilgilerini listeleyen sorgu

#### 3.10.1 Cevap-10

Bu SQL sorgusu şekil[14]'da, çalışanları maaşlarına göre birim bazında sıralar ve her bir birimde en yüksek maaşa sahip olan çalışanları belirler.

İlk olarak, RankedEmployees adlı bir tablo ifadesi (CTE - Common Table Expression) kullanılır. Bu tablo ifadesi, "calisanlar" tablosundan çalışanların adını, soyadını, maaşını, birim adını ve unvanını seçer. Ayrıca, ROW\_NUMBER() fonksiyonu kullanılarak her bir birim içinde maaşa göre sıralama yapılır. PARTITION BY ifadesi ile birimlere göre bölümlendirme yapılır ve ORDER BY ile maaşa göre sıralama yapılır.

Daha sonra, ana sorguda, RankedEmployees tablosundan, her bir birimde en yüksek maaşa sahip olan çalışanlar seçilir. WHERE koşulu ile sadece her birimdeki en yüksek maaşa sahip olan çalışanların sırası belirtilir (rank = 1).

Sonuç olarak, sorgu, her bir birimde en yüksek maaşa sahip olan çalışanların adını, soyadını, maaşını, birim adını ve unvanını döndürür.

The screenshot shows a SQL Developer window with two tabs: 'SQLQuery4.sql - DE...RESS.föy3 (sa (65))\*' and 'SQLQuery3.sql - DE...RESS.föy3 (sa (54))\*'. The active tab displays a SQL query that uses a CTE named 'RankedEmployees' to rank employees by salary within each department. The query selects the employee's name, salary, department name, and job title, ordered by department and then by salary in descending order. The results pane shows the top three highest-paid employees in each department.

```

WITH RankedEmployees AS (
    SELECT
        c.ad,
        c.soyad,
        c.maas,
        b.birim_ad,
        u.unvan_calisan,
        ROW_NUMBER() OVER (PARTITION BY c.calisan_birim_id ORDER BY c.maas DESC) AS rank
    FROM
        calisanlar c
    INNER JOIN
        birimler b ON c.calisan_birim_id = b.birim_id
    INNER JOIN
        unvan u ON c.calisan_id = u.unvan_calisan_id
)
SELECT
    ad,
    soyad,
    maas,
    birim_ad AS birim,
    unvan_calisan AS unvan
FROM
    RankedEmployees
WHERE
    rank = 1;

```

Results:

	ad	soyad	maas	birim	unvan
1	Ismail	Işeri	100000	Yazılım	Yönetici
2	Durmuş	Şahin	300000	Donanım	Takım Lideri
3	Kağan	Yazar	500000	Güvenlik	Yönetici Yardımcısı

Şekil 14: Cevap-10

## BÖLÜM: İİİ

### SONUÇ



---

## SONUÇ

---

Bu süreçte, öncelikle bir veritabanı oluşturuldu ve bu veritabanı içerisinde "calisanlar", "birimler", "ikramiye" ve "unvan" gibi tablolar tanımlandı. Tablolar arasında doğru ilişkiler kurularak veri bütünlüğü sağlandı. Daha sonra, her bir tabloya örnek veriler eklendi. Çalışanların ad, soyad, maaş gibi temel bilgileri veri tabanına girildi ve aynı zamanda birimleri, unvanları ve ikramiyeleriyle ilişkilendirildi.

Veri tabanı oluşturma işleminden sonra, çeşitli SQL sorguları kullanılarak veri sorgulamaları yapıldı. Bu sorgular aracılığıyla, en yüksek maaş alan çalışanların belirlenmesi, belirli bir maaş aralığındaki çalışanların listelenmesi, ikramiye hakkına sahip çalışanların bilgilerinin görüntülenmesi gibi çeşitli analizler gerçekleştirildi. Bu analizler, işletmenin personel politikalarını ve performansını anlamak için önemli bir rol oynadı.

Sonuç olarak, veri tabanı yönetimi ve SQL sorgulama konularında temel becerilerin geliştirilmesi ve bu becerilerin gerçek dünya senaryolarında nasıl uygulanacağını anlaşılması hedeflendi. Bu süreç, öğrencilerin veri tabanı yönetimi alanında pratik deneyim kazanmalarını ve veriye dayalı karar alma süreçlerine katkıda bulunabilecekleri bir temel oluşturmayı amaçlamıştır.

---

## KAYNAKÇA

---

- [1] Github: [https://github.com/aleykhrmn/veritabani\\_lab/tree/main/F%C3%B6y-3](https://github.com/aleykhrmn/veritabani_lab/tree/main/F%C3%B6y-3).