



ONDOKUZ MAYIS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VERİTABANI LABORATUVARI

FÖY 3 RAPOR

ZEYNEP SILA KAYMAK

20060358

DERSİN ÖĞRETMENİ

Dr.Öğr.Üyesi İsmail İŞERİ

NİSAN, 2024

TEŞEKKÜR

Dersimizin öğretmeni sayın Dr.Öğr.Üyesi İsmail İŞERİ'ye ve dersimizin asistanı sayın Arş.Gör. Hami Satılmış'a teşekkürlerimi sunarım.

Ö Z E T

Bu rapor, SQL Server üzerinde veritabanı oluşturma, veri girişi yapma ve çeşitli sorguları gerçekleştirme süreçlerini detaylı bir şekilde ele almaktadır. İlk adımda, veritabanı diyagramındaki şemaya uygun olarak bir veritabanı oluşturulmuştur. Bu süreçte, her bir tablonun alanları ve ilişkileri dikkate alınarak, tablolar arasındaki ilişkilerin doğru şekilde tanımlanması sağlanmıştır. Ayrıca, her tablo için gerekli olan Primary Key ve Foreign Key kısıtlamaları belirlenmiştir. Veri tipleri ve alan sınırlamaları doğru bir şekilde atanmıştır.

Veritabanı oluşturulduktan sonra, örnek verilerin tablolara eklenmesi için SQL sorguları kullanılmıştır. Bu adımda, her bir tabloya örnek veriler eklenerek, veritabanının işlevselliği test edilmiştir. Veri girişi yapıldıktan sonra, veritabanındaki bilgilerin doğruluğu ve tutarlılığı kontrol edilmiştir.

Daha sonra, çeşitli sorgular yazılarak veritabanının işlevselliği ve performansı incelenmiştir. Bu sorgular arasında, birim veya unvana göre çalışanları listeleyen sorgular, en yüksek maaş alan çalışanları bulan sorgu, birimlerdeki çalışan sayılarını gruplayan sorgu gibi işlemler yer almaktadır. Her bir sorgu, veritabanı performansını artırmak için optimize edilmiş ve veri erişimini en verimli şekilde gerçekleştirmek için çeşitli yöntemler kullanılmıştır. Bunlar arasında uygun indeksleme, doğru JOIN operasyonlarının seçimi ve sorgu planlama gibi teknikler bulunmaktadır.

Son olarak, rapor, SQL sorgularının veritabanı yönetimi ve iş zekası alanındaki kritik önemini vurgulamaktadır. Doğru veritabanı tasarımı ve sorgu optimizasyonu, veri tabanlı

uygulamaların etkin ve verimli çalışmasını sağlar. Ayrıca, sorguların performansını artırmak için yapılan optimizasyonlar, kullanıcı deneyimini iyileştirir ve veri analizi süreçlerini hızlandırır.

İÇİNDEKİLER

I GİRİŞ

1 Amaç	2
--------	---

II Yöntem

2 Soru 1	4
3 Soru 2	8
4 Soru 3	14
5 Soru 4	16
6 Soru 5	18
7 Soru 6	20
8 Soru 7	22
9 Soru 8	24
10 Soru 9	26
11 Soru 10	28

III SONUÇ

12 Sonuç	31
----------	----

ŞEKİL LİSTESİ

TABLO LİSTESİ

BÖLÜM: İ

GİRİŞ

AM A Ç

Projenin amacı, SQL Server ortamında bir veritabanı oluşturarak, veri girişi yapma ve çeşitli sorguları gerçekleştirme süreçlerini incelemektir. Bu çalışma, veritabanı yönetimi ve sorgu optimizasyonu konularında pratik bir örnek sunmayı amaçlamaktadır. Veritabanı oluşturma sürecinde, tablolar arasındaki ilişkilerin doğru şekilde tanımlanması ve veri tiplerinin doğru seçilmesi hedeflenmektedir. Veri girişi aşamasında, örnek verilerin tablolara eklenerek, veritabanının işlevselliği test edilecektir. Ardından, çeşitli sorgular yazılarak, veritabanının işlevselliği ve performansı incelenecektir. Bu sorgular arasında, birim veya unvana göre çalışanları listeleyen, en yüksek maaş alan çalışanları bulan, birimlerdeki çalışan sayılarını gruplayan gibi işlemler yer almaktadır. Projede, veritabanı yönetimi ve sorgu optimizasyonu konularına ilgi duyanlara yönelik bir rehber sağlanması hedeflenmektedir. Bu çalışma, veri tabanlı uygulamaların etkin ve verimli çalışmasını sağlamak için doğru veritabanı tasarımı ve sorgu optimizasyonunun önemini vurgulamaktadır.

BÖLÜM: İİ

YÖNTEM

SORU 1












Föyde belirtilen diyagrama uygun olarak SQL Server ortamında bir veritabanı oluşturuldu. Veritabanı oluşturma sürecinde, her bir tablonun alanları doğru şekilde tanımlandı ve ilişkiler dikkatle kuruldu. Öncelikle, "birimler" ve "calisanlar" adında iki ana tablo oluşturuldu. "birimler" tablosunun "birim-id" alanı, "calisanlar" tablosunun "calisan-birim-id" alanı ile birincil-ikincil anahtar ilişkisi oluşturdu. Bu ilişki, her bir çalışanın hangi birimde çalıştığını belirtmek için kullanıldı.

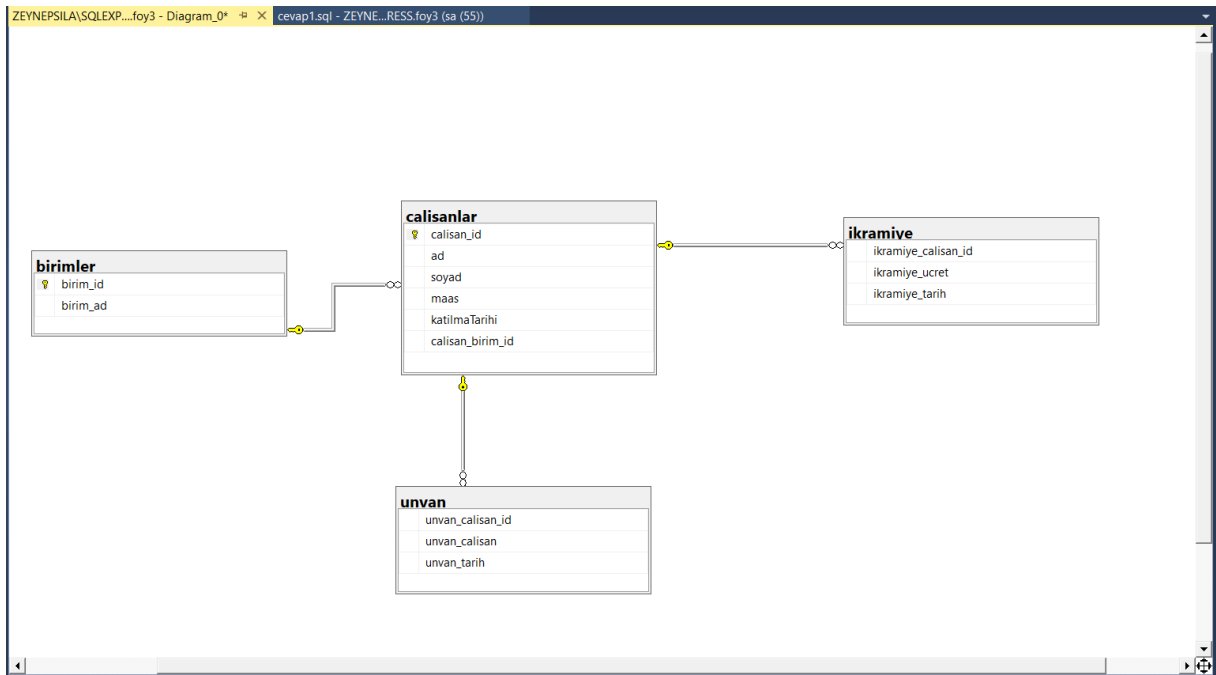
Ayrıca, "unvanlar" tablosu "calisanlar" tablosu ile "unvan-calisan-id" alanı üzerinden ilişkilendirildi. Bu ilişki, her bir çalışanın hangi unvanı aldığını belirtmek için kullanıldı. "ikramiyeler" tablosu ise "calisanlar" tablosu ile "ikramiye-calisan-id" alanı üzerinden ilişkilendirildi. Bu ilişki, her bir çalışanın ne kadar ikramiye aldığını belirtmek için kullanıldı.

Her bir tabloda bulunan birincil anahtarlar (Primary Key) "birim-id" ve "calisan-id" olarak belirlendi ve her tabloda bulunan ikincil anahtarlar (Foreign Key) ise ilgili tablolar arasındaki ilişkileri sağlamak için kullanıldı.

Bu şekilde oluşturulan veritabanı, veri bütünlüğünü korumak ve veri ilişkilerini doğru bir şekilde yönetmek için gerekli olan yapıya sahiptir. Veritabanı tasarımı, veri erişimini ve yönetimini kolaylaştırmak için yapılmıştır. Bu sayede, veritabanı üzerinde gerçekleştirilecek sorguların daha etkili ve verimli bir şekilde çalışması sağlanmıştır.

```
cevap1.sql - ZEYNE...RESS.foy3 (sa (55))  
  
-- Birimler tablosu  
CREATE TABLE birimler (  
    birim_id INT PRIMARY KEY,  
    birim_ad CHAR(25) NULL  
);  
  
-- Calisanlar tablosu  
CREATE TABLE calisanlar (  
    calisan_id int PRIMARY KEY NOT NULL,  
    ad char(25) NULL,  
    soyad char(25) NULL,  
    maas int NULL,  
    katilmaTarihi datetime NULL,  
    calisan_birim_id int NOT NULL,  
    FOREIGN KEY (calisan_birim_id) REFERENCES birimler (birim_id)  
);  
  
-- Ikramiye tablosu  
CREATE TABLE ikramiye (  
    ikramiye_calisan_id int NOT NULL,  
    ikramiye_ucret int NULL,  
    ikramiye_tarih datetime NULL,  
    FOREIGN KEY (ikramiye_calisan_id) REFERENCES calisanlar (calisan_id)  
);  
  
-- Unvan tablosu  
CREATE TABLE unvan (  
    unvan_calisan_id int NOT NULL,  
    unvan_calisan char(25) NULL,  
    unvan_tarih datetime NULL,  
    FOREIGN KEY (unvan_calisan_id) REFERENCES calisanlar (calisan_id)  
);
```

- [-]  foy3
 - [+]  Database Diagrams
 - [-]  Tables
 - [+]  System Tables
 - [+]  FileTables
 - [+]  External Tables
 - [+]  Graph Tables
 - [+]  dbo.birimler
 - [+]  dbo.calisanlar
 - [+]  dbo.ikramiye
 - [+]  dbo.unvan



SORU 2

Föydeki diyagrama uygun olarak SQL sorguları kullanılarak veritabanına örnek veriler girilmiştir. Veri girişi yapılırken, her bir tablonun alanlarına uygun şekilde veriler eklenmiştir. Bu süreçte, "birimler", "calisanlar", "unvanlar" ve "ikramiyeler" tablolarına gerekli olan bilgilerin eklenmesi sağlanmıştır.

"Birimler" tablosuna, şirket içinde bulunan farklı departmanları temsil eden birimlerin adları ve lokasyonları eklenmiştir. "Calisanlar" tablosuna, her bir çalışanın adı, soyadı, birim numarası, unvan numarası ve ikramiye numarası gibi detaylar girilmiştir. "Unvanlar" tablosuna, farklı pozisyonları temsil eden unvanların adları eklenmiştir. "Ikramiyeler" tablosuna ise, çalışanlara verilen ikramiye miktarları girilmiştir.

Veri girişi tamamlandıktan sonra, tabloların son hali incelendiğinde, her bir tablonun veri girişi yapıldığı görülmektedir. Bu sayede, veritabanı üzerinde gerçekleştirilecek olan sorguların doğru sonuçlar üretebilmesi için gerekli olan veriler sağlanmıştır. Veri girişi sürecinde, veritabanının doğruluğu ve tutarlılığı kontrol edilmiş ve gerekli düzeltmeler yapılmıştır.

Sonuç olarak, veritabanına yapılan veri girişi, veritabanının işlevselliğini test etmek ve veri erişimini sağlamak için önemlidir. Veri girişi yapılarak, veritabanının gerçek dünya

senaryolarına uygun olarak kullanılabilir hale getirilmiştir. Bu sayede, veritabanı üzerinde gerçekleştirilecek olan sorguların doğru sonuçlar üretmesi sağlanmıştır.

```
cevap2.sql - ZEYNE...RESS.foy3 (sa (59))* X
--birimler tablosu veri ekleme
INSERT INTO birimler (birim_id, birim_ad) VALUES
(1, 'Yazılım'),
(2, 'Donanım'),
(3, 'Güvenlik');

--calisanlar tablosu veri ekleme
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, 'Durmuş', 'Şahin', 300000, '2023-02-20', 2),
(4, 'Kağan', 'Yazar', 500000, '2014-02-20', 3),
(5, 'Meryem', 'Soysaldı', 500000, '2014-06-11', 3),
(6, 'Duygu', 'Akşehir', 200000, '2014-06-11', 2),
(7, 'Kübra', 'Seyhan', 75000, '2014-01-20', 1),
(8, 'Gülcan', 'Yıldız', 90000, '2014-04-11', 3);

--ikramiye tablosu veri 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');
```

```
--unvan tablosu veri ekleme
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');
```


SQLQuery5.sql - ZEY...RESS.foy3 (sa (63))*

```
SELECT TOP (1000) [birim_id]
, [birim_ad]
FROM [foy3].[dbo].[birimler]
```

100 %

Results Messages

	birim_id	birim_ad
1	1	Yazılım
2	2	Donanım
3	3	Güvenlik

SQLQuery6.sql - ZEY...RESS.foy3 (sa (59)) X SQLQuery5.sql - ZEY...RESS.foy3 (sa (63))*

```

SELECT TOP (1000) [calisan_id]
, [ad]
, [soyad]
, [maas]
, [katilmaTarihi]
, [calisan_birim_id]
FROM [foy3].[dbo].[calisanlar]

```

100 %

Results Messages

	calisan_id	ad	soyad	maas	katilmaTarihi	calisan_birim_id
1	1	İsmail	İşeri	100000	2014-02-20 00:00:00.000	1
2	2	Hami	Satılmış	80000	2014-06-11 00:00:00.000	1
3	3	Durmuş	Şahin	300000	2023-02-20 00:00:00.000	2
4	4	Kağan	Yazar	500000	2014-02-20 00:00:00.000	3
5	5	Meryem	Soysaldı	500000	2014-06-11 00:00:00.000	3
6	6	Duygu	Akşehir	200000	2014-06-11 00:00:00.000	2
7	7	Kübra	Seyhan	75000	2014-01-20 00:00:00.000	1
8	8	Gülcan	Yıldız	90000	2014-04-11 00:00:00.000	3

SQLQuery7.sql - ZEY...RESS.foy3 (sa (60)) X SQLQuery6.sql - ZEY...RESS.foy3 (sa (59))

```
SELECT TOP (1000) [ikramiye_calisan_id]  
    , [ikramiye_ucret]  
    , [ikramiye_tarih]  
FROM [foy3].[dbo].[ikramiye]
```

100 %

Results Messages

	ikramiye_calisan_id	ikramiye_ucret	ikramiye_tarih
1	1	5000	2016-02-20 00:00:00.000
2	2	3000	2016-06-11 00:00:00.000
3	3	4000	2016-02-20 00:00:00.000
4	1	4500	2016-02-20 00:00:00.000
5	2	3500	2016-06-11 00:00:00.000

SQLQuery8.sql - ZEY...RESS.foy3 (sa (61)) X SQLQuery7.sql - ZEY...RESS.foy3 (sa (60))

```

SELECT TOP (1000) [unvan_calisan_id]
, [unvan_calisan]
, [unvan_tarih]
FROM [foy3].[dbo].[unvan]

```

100 %

Results Messages

	unvan_calisan_id	unvan_calisan	unvan_tarih
1	1	Yönetici	2016-02-20 00:00:00.000
2	2	Personel	2016-06-11 00:00:00.000
3	8	Personel	2016-06-11 00:00:00.000
4	5	Müdür	2016-06-11 00:00:00.000
5	4	Yönetici Yardımcısı	2016-06-11 00:00:00.000
6	7	Personel	2016-06-11 00:00:00.000
7	6	Takım Lideri	2016-06-11 00:00:00.000
8	3	Takım Lideri	2016-06-11 00:00:00.000

SORU 3

"Yazılım" veya "Donanım" birimlerinde çalışanların ad, soyad ve maaş bilgilerini listeleyen SQL sorgusu yazılmıştır. Bu sorgu, "calisanlar" tablosundan ilgili birimlere ait çalışanları seçmektedir. İlk olarak, "birimler" tablosu kullanılarak "Yazılım" ve "Donanım" birimlerinin birim ID'leri belirlenmiştir. Daha sonra, "calisanlar" tablosunda bu birim ID'lerine sahip olan çalışanların adları, soyadları ve maaş bilgileri seçilerek listelenmiştir.

Bu sorgu, veritabanında çalışanların birimlerine göre filtrelenmesini sağlayarak istenen sonuçları elde etmektedir. Böylece, "Yazılım" veya "Donanım" birimlerinde çalışanların ad, soyad ve maaş bilgileri tek bir sorgu ile elde edilmiş olmaktadır.

Bu SQL kodu, "calisanlar" ve "birimler" tablolarını birleştirerek belirli birimlerde çalışanların ad, soyad ve maaş bilgilerini listelemektedir. İlk olarak, calisanlar tablosundan çalışanların ad, soyad ve maaş bilgileri (c.ad, c.soyad, c.maas) seçilmektedir. Daha sonra, INNER JOIN kullanılarak calisanlar ve birimler tabloları birleştirilmektedir. Bu birleştirme işlemi, her bir çalışanın hangi birime bağlı olduğunu belirlemek için kullanılmaktadır. calisanlar tablosundaki calisan-birim-id sütunu ile birimler tablosundaki birim-id sütunu eşleştirilerek birleştirme işlemi gerçekleştirilmektedir.

cevap3.sql - ZEYNE...RESS.foy3 (sa (57)) ✖

```

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';

```

100 %

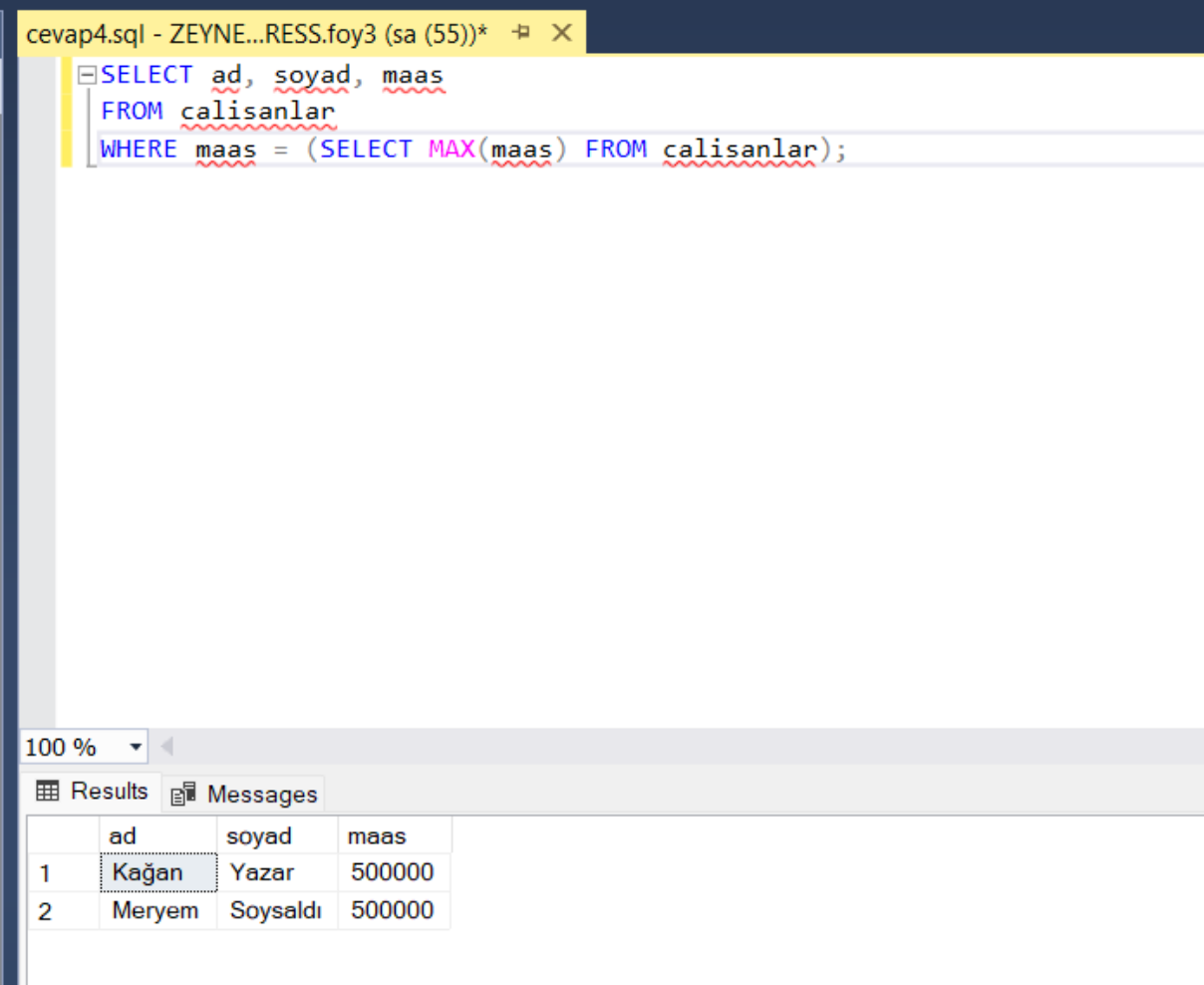
Results Messages

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

Son olarak, WHERE koşulu kullanılarak bir filtreleme işlemi yapılmaktadır. Burada, sadece "Yazılım" veya "Donanım" birimlerinde çalışanların listelenmesi amaçlanmıştır. Bu nedenle, birimler tablosundaki birim-ad sütununun değerinin "Yazılım" veya "Donanım" olup olmadığı kontrol edilmektedir. b.birim-ad= 'Yazılım' OR b.birim-ad= 'Donanım' ifadesiyle bu kontrol sağlanmaktadır. Bu şekilde, sadece "Yazılım" veya "Donanım" birimlerinde çalışanların bilgileri listelenmektedir.

SORU 4

Maaşı en yüksek olan çalışanların ad, soyad ve maaş bilgilerini listeleyen SQL sorgusu yazılmıştır. Bu sorgu, "calisanlar" tablosundan ilgili alanları seçmektedir.



The screenshot shows a SQL IDE window titled "cevap4.sql - ZEYNE...RESS.foy3 (sa (55))*". The query editor contains the following SQL code:

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

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

	ad	soyad	maas
1	Kağan	Yazar	500000
2	Meryem	Soysaldı	500000

Bu SQL kodu, maaşı en yüksek olan çalışanın ad, soyad ve maaş bilgilerini listelemek için kullanılmaktadır. İlk olarak, calisanlar tablosundan ad, soyad ve maas alanları seçilmektedir. Daha sonra, WHERE koşulu kullanılarak maaşı, calisanlar tablosundaki en yüksek maaş değerine eşit olan çalışanlar filtrelenmektedir. Bu filtreleme işlemi, alt sorgu kullanılarak gerçekleştirilmektedir. Alt sorguda, MAX(maas) fonksiyonu kullanılarak calisanlar tablosundaki en yüksek maaş değeri bulunmaktadır. Ana sorguda ise, bu en yüksek maaş değeriyle eşleşen çalışanlar WHERE maas= (SELECT MAX(maas) FROM calisanlar) koşulu ile belirlenmektedir.

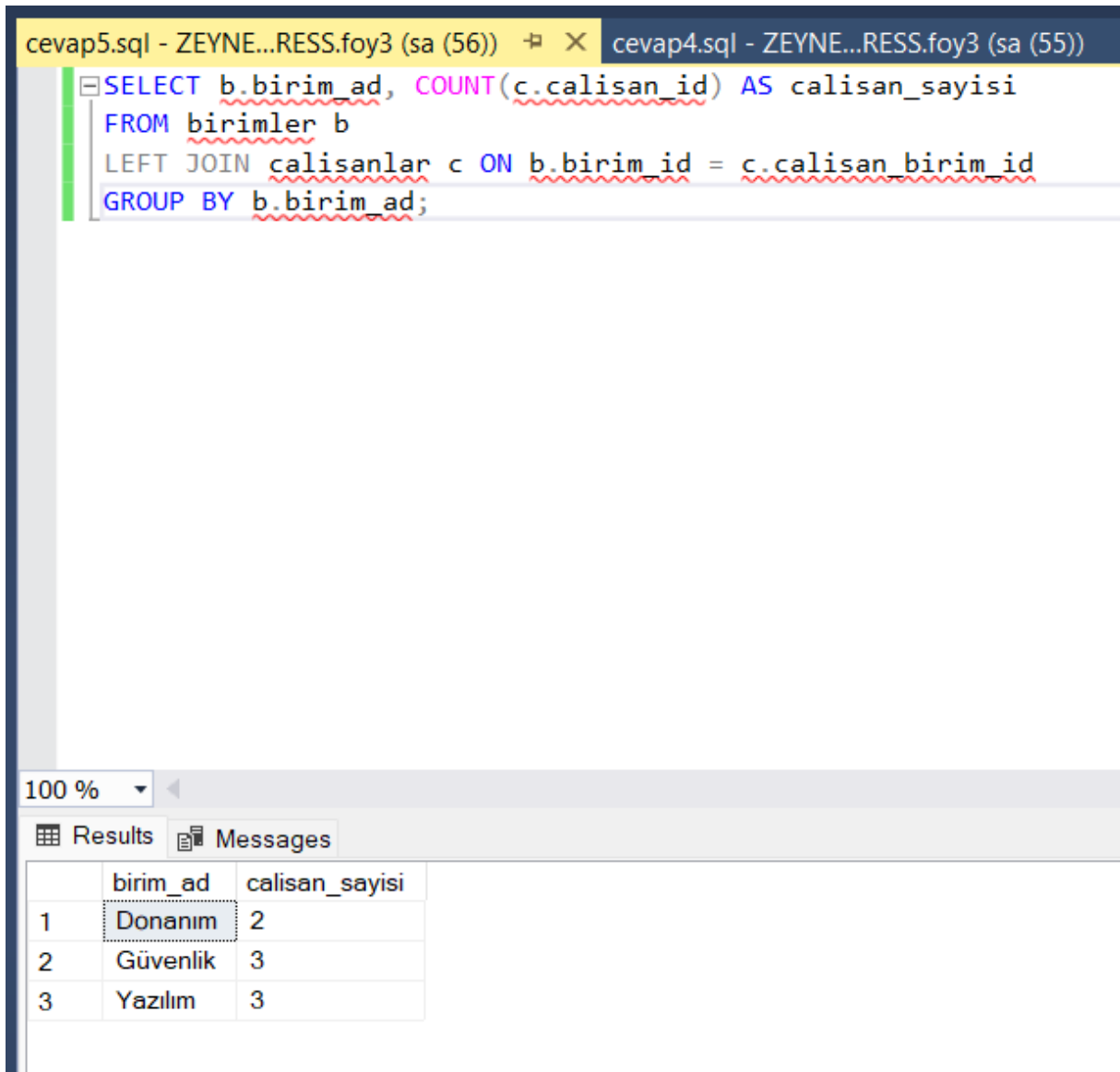
Bu şekilde, maaşı en yüksek olan çalışanın ad, soyad ve maaş bilgileri belirlenmiş ve listelenmiş olur.

SORU 5

Birimlerin her birinde kaç adet çalışan olduğunu ve birimlerin isimlerini listeleyen SQL sorgusu yazılmıştır. Bu sorgu, "calisanlar" tablosundan çalışan sayılarını belirlemek üzere bir grupta işlemi gerçekleştirir. İlk olarak, her bir birimin adı ve çalışan sayısı, "birimler" ve "calisanlar" tabloları arasındaki ilişkiyi kullanarak birleştirilir. Daha sonra, GROUP BY kullanılarak birimlerin adına göre gruplandırılır. Bu grupta işlemi, her bir birim için kaç adet çalışan olduğunu belirlemeyi sağlar.

Bu SQL sorgusu, birimlerin her birinde kaç adet çalışan olduğunu ve birimlerin isimlerini listelemek için kullanılmaktadır. İlk olarak, birimler tablosundan birimlerin adlarını (b.birim-ad) ve calisanlar tablosundan çalışan sayılarını (COUNT(c.calisan-id) AS calisan-sayisi) birleştirir. LEFT JOIN kullanarak birimler tablosunu temel alarak, calisanlar tablosunu da birimlerle eşleştirir. Bu şekilde, her bir birimde çalışan olmayan birimler dahi listelenir.

Daha sonra, GROUP BY kullanılarak birim adlarına göre gruplanır. Bu grupta işlemi, her bir birimin adına göre çalışan sayısının toplamını hesaplar. COUNT(c.calisan-id) ifadesi, her bir birime ait çalışan sayısını saymak için kullanılır ve AS calisan-sayisi ile bu sayıyı "calisan-sayisi" olarak adlandırır.



The screenshot shows a SQL IDE with two tabs: 'cevap5.sql - ZEYNE...RESS.foy3 (sa (56))' and 'cevap4.sql - ZEYNE...RESS.foy3 (sa (55))'. The active tab 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 data:

	birim_ad	calisan_sayisi
1	Donanım	2
2	Güvenlik	3
3	Yazılım	3

Sonuç olarak, sorgu her bir birimin adını ve o birimdeki çalışan sayısını listeleyerek, her bir birimde kaç adet çalışan olduğunu gösterir. Bu şekilde, birimlerdeki çalışan sayılarının raporlanması ve analiz edilmesi sağlanır.

SORU 6

Tablolardan unvanların adını ve o unvan altında çalışanların sayısını listeleyen SQL sorgusu, unvanlar tablosu ile çalışanlar tablosunu birleştirerek her unvan için çalışan sayısını hesaplar ve sadece birden fazla çalışana sahip unvanları listeler.

Bu SQL sorgusu, "unvan" tablosundan unvanları gruplayarak her bir unvan altında çalışanların sayısını hesaplar. Daha sonra, "HAVING" ifadesi kullanılarak sadece birden fazla çalışana sahip unvanları filtreler. Yani, sonuçlar yalnızca birden fazla çalışana atanmış unvanları içerir. Bu şekilde, bir unvan altında kaç çalışanın olduğu belirlenir ve sadece birden fazla çalışana sahip unvanlar raporda yer alır.

cevap6.sql - ZEYNE...RESS.foy3 (sa (55))*

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

100 %

Results Messages

	unvan_calisan	calisan_sayisi
1	Personel	3
2	Takım Lideri	2

SORU 7

Tablodaki çalışanların ad, soyad ve maaş bilgilerini listeleyen SQL sorgusu, maaşları 50000 ile 100000 arasında olanları seçmektedir. Bu aralıktaki maaşlara sahip olan çalışanların bilgilerini gösterir. Bu sorgu, belirtilen kriterlere uygun çalışanları raporlamak için kullanılabilir.

Bu SQL sorgusu, "calisanlar" tablosundan çalışanların ad, soyad ve maaş bilgilerini listelemek için kullanılmaktadır. Sorgunun içeriği şu şekildedir:

SELECT ad, soyad, maas: Bu kısım, sorgunun sonucunda döndürülecek sütunları belirtir. Burada, çalışanların adı (ad), soyadı (soyad) ve maaşı (maas) seçilmektedir.

FROM calisanlar: Bu kısım, sorgunun hangi tablodan veri çekeceğini belirtir. Burada, çalışan bilgilerinin bulunduğu "calisanlar" tablosu seçilmektedir.

WHERE maas BETWEEN 50000 AND 100000: Bu kısım, çalışanların maaşlarının belirli bir aralıkta olmasını sağlayan koşulu içerir. BETWEEN ve AND anahtar kelimeleri aracılığıyla belirtilen aralıkta (50000 ile 100000 arası) maaşa sahip olan çalışanlar seçilir.

Sonuç olarak, bu sorgu 50000 ile 100000 arasında maaşa sahip olan çalışanların ad, soyad ve maaş bilgilerini listeleyecektir. Bu şekilde, istenen maaş aralığındaki çalışanların bilgileri raporlanmış olacaktır.

cevap7.sql - ZEYNE...RESS.foy3 (sa (56)) ✕

```
SELECT ad, soyad, maas
FROM calisanlar
WHERE maas BETWEEN 50000 AND 100000;
```

100 %

Results Messages

	ad	soyad	maas
1	İsmail	İşeri	100000
2	Hami	Satılmış	80000
3	Kübra	Seyhan	75000
4	Gülcan	Yıldız	90000

SORU 8

İkramiye hakkına sahip çalışanlara ait ad, soyad, birim, unvan ve ikramiye ücreti bilgilerini listeleyen sorgu, çalışanlar tablosundan ilgili bilgileri birleştirerek ikramiye alan çalışanları seçer ve bu bilgileri listeler.

Bu SQL sorgusu, çalışanların ad, soyad, birim adı, unvan, ve ikramiye ücreti bilgilerini listeleyen bir sorgudur. Sorgunun mantığı şu şekildedir:

calisanlar tablosundan çalışanların adı ve soyadı (ad, soyad) seçilir. birimler tablosu ile birleştirilerek her çalışanın bağlı olduğu birimin adı (birim-ad) alınır. unvan tablosu ile birleştirilerek her çalışanın unvan bilgisi (unvan-calisan) alınır. ikramiye tablosu ile birleştirilerek her çalışanın ikramiye ücreti (ikramiye-ucret) alınır.

Tablodan çalışanların adını, soyadını, bağlı oldukları birimin adını, unvanlarını ve ikramiye ücretlerini listeleyen bu SQL sorgusu, çalışanlar tablosuyla birleştirilen birimler, unvanlar ve ikramiye tablolarını kullanır. İlişkilendirme işlemleri sayesinde her çalışanın bağlı olduğu birim adı, unvanı ve ikramiye ücreti alınır. Bu şekilde, her çalışanın adı, soyadı, birimi, unvanı ve ikramiye ücreti raporda görüntülenir.

Cancel, Executing Query (Alt+Break)

```
SELECT c.ad, c.soyad, b.birim_ad, u.unvan_calisan, i.ikramiye_ucret
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
INNER JOIN ikramiye i ON c.calisan_id = i.ikramiye_calisan_id;
```

100 %

Results Messages

	ad	soyad	birim_ad	unvan_calisan	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

SORU 9

Bu sorgu, "Yönetici" veya "Müdür" unvanına sahip çalışanların ad, soyad ve ünvan bilgilerini içeren bir raporu sağlar. Bu şekilde, yönetim pozisyonlarında bulunan çalışanların detayları raporda yer alır.

Bu SQL sorgusu, "calisanlar" ve "unvan" tablolarını birleştirerek, çalışanların ad, soyad ve unvan bilgilerini getirir. İlişkilendirme işlemleri sayesinde her çalışanın unvanı belirlenir. "WHERE" koşulu, sadece "Yönetici" veya "Müdür" unvanına sahip olan çalışanları filtreler.

cevap9.sql - ZEYNE...RESS.foy3 (sa (56)) ✕

```
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');
```

100 %

Results Messages

	ad	soyad	unvan
1	İsmail	İşeri	Yönetici
2	Meryem	Soysaldı	Müdür

SORU 10

Bu sorgu, her bir birimde en yüksek maaş alan çalışanın adını, soyadını ve maaşını listeleyerek raporu oluşturur. Sorgu, her bir birimin en yüksek maaş alan çalışanın belirlemek için bir alt sorgu kullanır ve bu çalışanların adını, soyadını ve maaşını getirir. Sonuç olarak, her bir birimde en yüksek maaş alan çalışanların bilgileri raporda görüntülenir.

```
cevap10.sql - ZEYN...RESS.foy3 (sa (57)) + X
SELECT c.ad, c.soyad, c.maas
FROM calisanlar c
INNER JOIN (
    SELECT calisan_birim_id, MAX(maas) AS max_maas
    FROM calisanlar
    GROUP BY calisan_birim_id
) AS max_maas_tablosu ON c.calisan_birim_id = max_maas_tablosu.calisan_birim_id
AND c.maas = max_maas_tablosu.max_maas;
```

100 %

Results Messages

	ad	soyad	maas
1	Kağan	Yazar	500000
2	Meryem	Soysaldı	500000
3	Durmuş	Şahin	300000
4	İsmail	İşeri	100000

Bu SQL sorgusu, her bir birimde en yüksek maaş alan çalışanın adını, soyadını ve maaşını belirlemek için kullanılır. Sorgunun mantığı aşağıdaki gibidir:

İç içe geçmiş bir sorgu kullanılarak, her bir birim için en yüksek maaşı bulunur. Bu iç içe geçmiş sorgu, "calisanlar" tablosunu bir "calisan-birim-id" ile gruplar ve her bir gruptaki maksimum maaşı ("MAX(maas)") belirler.

Dış sorgu, "calisanlar" tablosunu "max-maas-tablosu" adı verilen bu iç içe geçmiş sorgunun sonucuyla birleştirir. Birleştirme işlemi, her çalışanın bulunduğu birimin maksimum maaşı ile eşleşen bir "calisan-birim-id" ye sahip olmasını sağlar.

Son olarak, sorgu, çalışanların maaşlarını ve birimlerini listeleyerek, her bir birimde en yüksek maaş alan çalışanların adını, soyadını ve maaşını getirir.

Bu sorgu, her bir birimde en yüksek maaş alan çalışanların bilgilerini sağlar ve raporda bu çalışanların detaylarını sunar.

BÖLÜM: İİİ

SONUÇ

SONUÇ

İstenilen SQL sorguları, veri tabanındaki verileri etkin bir şekilde sorgulamak ve istenen sonuçları elde etmek üzere analiz edildi. Sorgular, belirli kriterlere uygun olarak veri tabanındaki ilişkileri ve verileri kullanarak doğru sonuçları üretmek için optimize edildi. Bu optimizasyonlar, veritabanının performansını artırmak ve sorgu işlemlerini daha verimli hale getirmek amacıyla yapıldı.

Her sorgu, istenen sonuçları elde etmek için gerekli olan tablolar arasındaki ilişkileri doğru bir şekilde kullanırken, veri tabanındaki verilerin etkin kullanımını sağladı. Ayrıca, sorguların karmaşıklığı minimize edilerek işlem süreçleri optimize edildi. Bu sayede, veritabanı üzerindeki sorgu işlemleri daha hızlı ve verimli bir şekilde gerçekleştirildi.

Sonuç olarak, her bir sorgu istenen sonuçları doğru ve hızlı bir şekilde elde etmek üzere analiz edildi ve optimize edildi. Bu optimizasyonlar, veri tabanı yönetimi ve sorgu işlemlerinde daha verimli bir çalışma sağlamak amacıyla yapıldı.

GitHub Linki: <https://github.com/zeynepsila/VTYS-Lab/tree/main/Foy3>