T.C.
ONDOKUZ MAYIS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Veri Tabanı Laboratuvarı

Deney Föyü-3

SORGU ANALİZİ YAPMA VE SORGU OPTİMİZASYONU

SQL (Structured Query Language) SORGULARI

Bazı sorgu örnekleri aşağıdaki "musteriler" veritabanı tablosuna göre verilmektedir.

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
3	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20
4	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40
5	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84

"musteriler" tablosu

SELECT

Veritabanındaki tablo veya tablolardan istenilen özellikteki verileri seçip listeleme için kullanılan komuttur.

Basit Kullanımı:

SELECT [sutun_adi] FROM [tablo_adi]

Temel Yapısı:

SELECT [ALL] [DISTINCT] <sütun (lar)> FROM [tablo_adi]

[WHERE ifade]

[GROUP BY ifade]

[HAVING ifade]

[ORDER BY ifade]

Örnek: SELECT * FROM [tablo_adi]

Yukarıdaki deyim ile tablodaki bütün veriler elde edilir. SELECT deyiminin ardından kullanılan * (asterisk) işareti bütün sütunlar anlamına gelmektedir.

Örnek: Müşteriler tablosunda müşterilerin adını ve soyadını listelemek için aşağıdaki sorgu yazılmaktadır.

```
SELECT ad, soyad FROM musteriler
```

	ad	soyad
1	İsmail	İşeri
2	Hami	Satılmış
3	Kağan	Yazar
4	Meryem	Soysaldı
5	Durmuş	Şahin

ALL

Bütün satırların listelenmesini sağlamaktadır.

Kullanımı:

SELECT ALL sutun_adi FROM tablo_adi

Örnek: Müşteriler tablosundaki sehir sütunundaki tüm verileri(satırları) listelemek için;

```
SELECT ALL sehir FROM müşteriler
```

	sehir
1	Tokat
2	Samsun
3	Samsun
4	Nevşehir
5	Mersin

DISTINCT

Birbirleriyle aynı olan satırların, listeleme esnasında bir kez yazılmasını sağlayan ifadedir.

Kullanımı: .
SELECT DISTINCT sutun_adi FROM tablo adı

Örnek: Müşteriler tablosundaki şehirlerin listesini tekrar etmeden almak için;

```
SELECT DISTINCT sehir FROM musteriler
```

	sehir
1	Mersin
2	Nevşehir
3	Samsun
4	Tokat

ORDER BY

Listelenecek bilgilerin belirli bir sütun adına göre sıralanmasını sağlamak için kullanılan komuttur.

Kullanımı:

SELECT * FROM tablo adi ORDER BY sutun_adi ASC

- **ASC:** Artan düzende kayıtları sıralar
- **DESC:** Azalan düzende kayıtları sıralar

Ayrıca tablo içindeki veriler sıralanırken aynı anda birden fazla alana göre sıralama yapmak mümkündür.

Örnek: Müşteriler tablosundaki verileri ad sütununa göre artan ve azalan olarak sıralamak için (Alfabetik sıralama);

```
SELECT * FROM musteriler ORDER BY ad ASC
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
3	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
4	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20
5	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40

```
SELECT * FROM musteriler ORDER BY ad DESC
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40
2	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20
3	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
4	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
5	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84

Karşılaştırma Operatörleri

(NOT, OR, AND), birden fazla koşula göre sıralama işlemlerinde bu ifadeler kullanılır.

Operatör	Anlamı
>	Büyük
<	Küçük
=	Eşit
<=	Küçük Eşit
>=	Büyük Eşit
<>	Eşit Değil

WHERE

Veri tabanında veriyi alma işlemi sırasında satırlara birtakım sınırlamalar getirilerek tablonun tüm satırları yerine istenildiği kadarını elde etmek mümkündür. Tabloda belirli kısımları seçme işlemini gerçekleştirmek için “WHERE” sözcüğü kullanılmaktadır.

Nümerik ifadelerde Kullanımı:

```
SELECT * FROM tablo_adi WHERE sutun_adi > 60
```

Karakterlerde ifadelerde Kullanımı:

```
SELECT * FROM tablo_adi WHERE sutun_adi = ‘murat’
```

Tarih türü ifadelerde Kullanımı:

```
SELECT * FROM tablo_adi WHERE sutun_adi > ’1979/09/19’
```

Örnek: Müşteriler tablosunda puanı 60'dan yüksek olan kayıtları döndürmek için;

```
SELECT * FROM musteriler WHERE puan > 60
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
3	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84

BETWEEN

Belirli bir aralık içindeki bilgileri listelemek için kullanılır.

Kullanımı:

SELECT * FROM tablo_adi WHERE sutun_adi BETWEEN baslangic AND bitiş

Örnek: Müşteriler tablosunda puanları 40 ile 65 arasındaki kayıtları döndürmek için;

```
SELECT * FROM musteriler WHERE puan BETWEEN 40 AND 65
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
2	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40

LIKE

Eğer aradığımız kayıtın bulunması için tam bir karşılaştırma yapamıyorsak o zaman kullanırız. Örnek olarak "müşteriler" tablosunda adının baş harfi "K" ile başlayan kayıtları bulmak için

- `SELECT * FROM musteriler WHERE ad LIKE 'K%'`

ifadesi kullanılır.

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20

"%" işareti uzunluğu önemsiz olmak üzere yazıldığı yere her türlü ifade gelebilir anlamındadır.

Aritmetik İşlemler

SELECT komutu ile veri tabanında mevcut tablolardan listeleme yaparken, tabloda ayrı bir sütun olarak yer almamış ve bir hesaplama sonucunda üretilebilecek bilgileri de liste içine katmak mümkündür.

Örnek: Müşteriler tablosunda şu anda geçerli olan puan ve bu puanın %10 fazla olan halini listelemek için;

```
SELECT puan, puan*1.1 FROM müşteriler
```

	puan	(No column name)
1	75	82.5
2	61	67.1
3	20	22.0
4	40	44.0
5	84	92.4

Kümeleme Fonskiyonları

SQL tablo içerisindeki çeşitli matematiksel ifadelerin sonucunu otomatik olarak üretmeyi sağlayan fonksiyonlara sahiptir.

SUM Fonksiyonu: Tablo içerisinde belirli bir sütuna göre toplama işlemi gerçekleştirmektedir.

Kullanımı:

SELECT SUM sutun_adi FROM tablo_adi;

Örnek: Müşteriler tablosunda puan sütunundaki tüm puanların toplamını döndürmek için;

```
SELECT SUM(puan) FROM müşteriler
```

	(No column name)
1	280

AVG Fonksiyonu: Belirli bir alan üzerinde aritmetik ortalama (avarage) hesaplamak için kullanılmaktdır.

Kullanımı:

SELECT AVG sutun_adi FROM tablo_adi

Örnek: Müşteriler tablosunda puan sütunundaki tüm puanların ortalamasını döndürmek için;

```
SELECT AVG(puan) FROM musteriler
```

	(No column name)
1	56

MAX ve MIN Fonksiyonu: İçinde, belirlenen sütun içindeki en büyük değeri bulmak için “MAX”, en küçük değeri bulmak için “MIN” fonksiyonu kullanılmaktadır.

Kullanımı:

SELECT MAX sutun_adi FROM tablo_adi
SELECT MIN sutun_adi FROM tablo_adi

Örnek: : Müşteriler tablosunda en düşük puanı bulmak için;

```
SELECT MIN(puan) FROM müşteriler
```

	(No column name)
1	20

COUNT Fonksiyonu: Tablo içerisinde herhangi bir alanda sayma işlemi gerçekleştirir.

Kullanımı:

SELECT COUNT sutun_adi FROM tablo_adi

Örnek: Müşteriler tablosunda cinsiyeti kadın olan verilerin sayısını bulmak için;

```
SELECT COUNT(cinsiyet) FROM musteriler where cinsiyet='K'
```

	(No column name)
1	1

GROUP BY: Kümeleme fonksiyonları kullanılırken tablodaki bilgileri, bazı özelliklere göre gruplandırarak bu gruplandırılmış veri üzerinde sorgulama yapmak mümkündür. Bu işlem için, GROUP BY ifadesi kullanılmaktadır.

Kullanımı:

SELECT gruplanan_sutun_adi, AVG(sutun_adi) WHERE tablo_adi GROUP BY gruplanan_sutun_adi

Örnek: Müşteriler tablosunda puanların ortalamasını cinsiyete göre bulmak için;

```
SELECT cinsiyet,AVG(puan) FROM musteriler GROUP BY cinsiyet
```

	cinsiyet	(No column name)
1	E	60
2	K	40

HAVING: Gruplandırarak, kümele fonksiyonlarını uygularken, koşul da verilebilir. Bu durumda, grup üzerindeki hesaplamalarla ilişkili koşul belirtirken, HAVING sözcüğünü kullanmak gerekmektedir.

Kullanımı:

SELECT sutun_adi FROM tablo_adi GROUP BY sutun_adi HAVING sutun_adi

Örnek: Müşteriler tablosunda puanı 40'dan yüksek olup, ortalama puanları 50'ye eşit ya da büyük olan müşterilerin adlarını listelemek için;

```
SELECT ad FROM musteriler WHERE puan > 40 GROUP BY ad HAVING AVG(puan) >= 50
```

	ad
1	Durmuş
2	Hami
3	İsmail

ALT SORGULAR

Bazı durumlarda bir sorgudan elde edilen sonuç diğer başka bir sorgu içerisinde kullanılabilmektedir. Bu tür durumlarda iç içe sorgular oluşturulmaktadır. Kullanılan iç içe sorgularda yer alan içteki sorgulara "alt sorgular" adı verilir.

1. Alt Sorgu Düzenleme Kuralları: Alt sorgular düzenlenirken uyulması gereken birtakım kurallar bulunmaktadır. Bunlar;

- FROM sözcüğü içinde tanımlanan sorgular dışında, alt sorgu, ana sorgu içerisindeki karşılaştırma işlecinin sağ tarafında yer almalıdır.
- Alt sorgu, parantezler içerisinde yer almalıdır.
- Alt sorgu ORDER BY sözcüğünü içermemelidir. ORDER BY sadece ana sorgu içerisinde yer alabilmektedir.

2. Alt Sorgunun Tanımlanması: Alt sorgu bir SELECT, SELECT...INTO, INSERT...INTO, DELETE veya UPDATE deyimi içinde veya başka bir alt sorguda SELECT deyiminin kullanılması ile elde edilir.

Kullanımı: SELECT sutun_adi FROM tablo_adi WHERE ifade karşılaştırma (SELECT sutun_adi FROM tablo_adi)

İÇ İÇE SELECT SORGULARI: Bazı sorgulamalar, özelliği itibarı ile "İÇ İÇE SELECT" komutlarının kullanılmasını gerektirebilir. İçteki "SELECT" komutunun bulduğu sonuç, dıştaki "SELECT" komutunun işlevini yerine getirebilmesi için kullanılmaktadır.

Kullanımı:

SELECT * FROM tabo1 WHERE sutun1= (SELECT sutun2 FROM tablo2 WHERE sutun2=deger)

Örnek: Müşteriler tablosunda puanı ortalama puandan büyük olan müşterileri listelemek için;

```
SELECT * FROM musteriler WHERE puan > (SELECT AVG(puan) FROM musteriler)
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
3	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84

IN ifadesi: Belirtilen değerlerden herhangi birine eşit olan kayıtları listelemeyi sağlamaktadır.

Örnek: Müşteriler tablosunda şehri Samsun ve Nevşehir olan kayıtları listelemek için;

```
SELECT * FROM musteriler WHERE sehir IN ('Samsun','Nevşehir')
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
2	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20
3	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40

ANY İfadesi:

- **>Any** komutu verilen değerlerin herhangi birinden büyük olan (en küçüğünden büyük) kayıtları listelemeyi sağlar.
- **<Any** komutu ise verilen değerlerin herhangi birinden küçük olan (en büyüğünden küçük) kayıtları listelemeyi sağlar.
- **=Any** komutu ise verilen değerlerden herhangi birine eşit olan kayıtları listelemeyi sağlar. **>=Any** ve **<=Any** gibi de kullanılabilir.

Örnek: Müşteriler tablosunda, Samsun şehrinde en düşük puanlı müşteriden daha fazla puana sahip müşterileri listelemek için;

```
SELECT * FROM musteriler WHERE puan > ANY (SELECT puan FROM musteriler WHERE sehir='Samsun')
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
3	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84

ALL İfadesi: Belirtilen şartların tamamının sağlanması istendiğinde kullanılmaktadır.

- **>All** komutu verilen değerlerin hepsinden büyük olan (en büyüğünden büyük) kayıtları listelemeyi sağlar.
- **<All** komutu ise verilen değerlerin hepsinden küçük olan (en küçüğünden küçük) kayıtları listelemeyi sağlar.
- **>=All** ve **<=All** gibi kullanımları da vardır.

Örnek: Müşteriler tablosunda, Samsun şehrinde en yüksek puanlı müşteriden daha fazla puana sahip müşterileri listelemek için;

```
SELECT * FROM musteriler WHERE puan > ALL (SELECT puan FROM musteriler WHERE sehir='Samsun')
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
2	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84

TABLOLARIN BİRLEŞTİRİLMESİ

Birden fazla tablodan veri almak gerektiği durumlarda tablolar arasında ilişki kurulması gerekmektedir. Bu işleme **JOIN** (birleştirme) adı verilir. Join işlemi birden fazla tabloyu birbirine bağlayıp bu tablolar üzerinde işlem yapabilmemizi sağlamaktadır. Join başka bir ifadeyle, table ve view gibi kaynakları ilişkilendiren, SELECT, INSERT INTO SELECT, SELECT INTO, UPDATE ve DELETE gibi işlemlerde FROM ile birlikte kullanılan ve bu şekilde tabloları (iki veya daha fazla) birleştiren ve sorgular temelinde bir sonuç tablosu (result table) oluşturmaya yarayan bir birleştiricidir. Birleştirme işlemi yapabilmek için tabloların aynı değerlerini içeren sütunlarının kullanılması gerekmektedir.

Basit Kullanımı:

SELECT * **FROM** [table1] **JOIN** [table2] **ON** [table1.primary_key] = [table2.foreign_key];

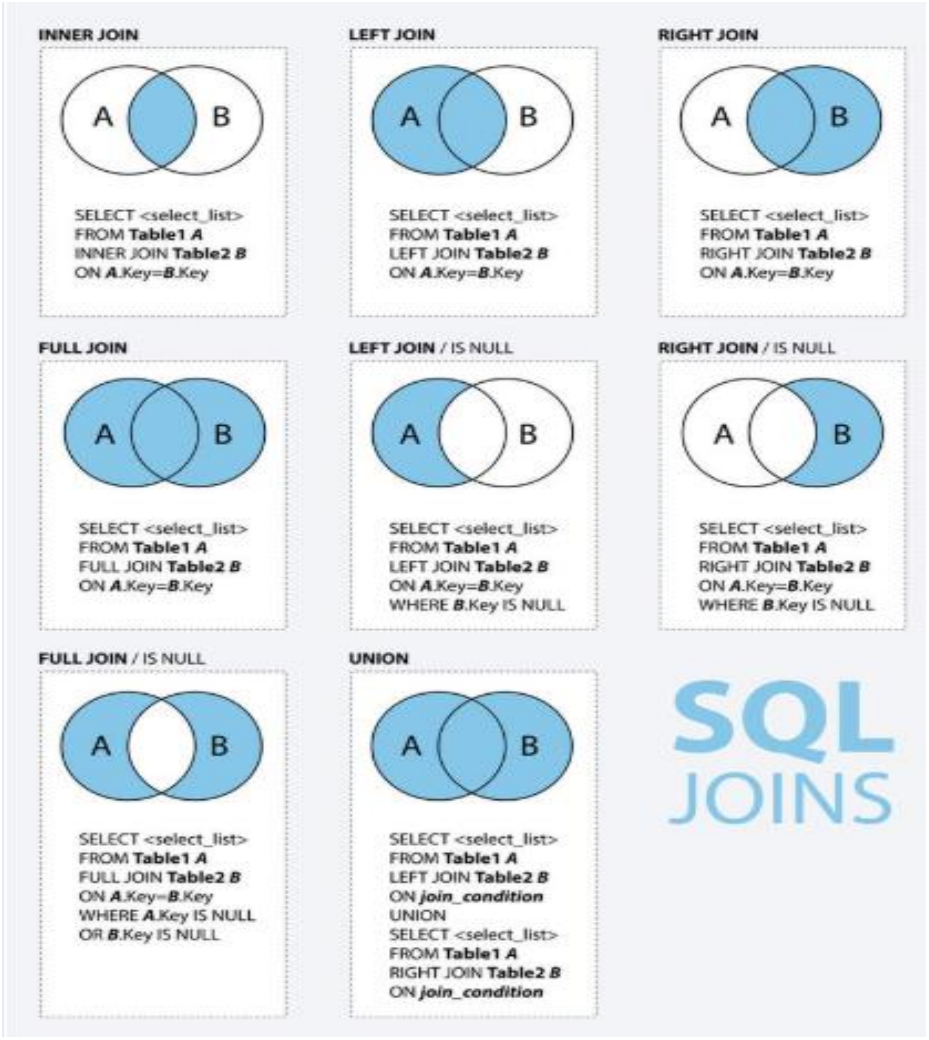
Join'in basit kullanımına bakacak olursak, **JOIN** ile hangi tabloların birleştirileceği belirtilmektedir; **table1** ve **table2. ON** ifadesi ise bu tabloların hangi alanlar bağlamında ilişkilendirileceğini belirtmektedir; **table1.primary_key** ile **table2.foreign key**.

JOIN Çeşitleri

JOIN sorguları, verileri elde etmek istediğimiz şekle göre aşağıdakiler gibi olabilirler:

- INNER JOIN (İç Birleştirici)
- OUTER JOIN (Dış Birleştirici)
- LEFT OUTER JOIN veya LEFT JOIN
- RIGHT OUTER JOIN veya RIGHT JOIN
- FULL JOIN veya FULL OUTER JOIN
- CROSS JOIN

Aşağıdaki görselde JOIN çeşitlerinin kümeler üzerinde nasıl işleyişe sahip oldukları gösterilmektedir.



JOIN Çeşitleri

JOIN çeşitlerinin işlevlerini açıklamadan önce örneklerde kullanılacak veritabanı örneğinin diyagramı ve tablolardaki veriler aşağıda verilmektedir.

Örnek Veritabanı Diyagramı

	kulup_no	kulup_ad
1	3	Robotik
2	1	Siber Güvenlik
3	2	Yapay Zeka

	ogrenci_no	ad	kulup_no
1	1	İsmail	1
2	2	Hami	1
3	3	Meryem	2
4	4	Durmuş	2
5	5	Kağan	NULL

Örnek Veritabanındaki Kulüp ve Öğrenci Tabloları

INNER JOIN: Varsayılan seçenektir ve birebir ilişki için tercih edilir. İki veya daha fazla veri tablosunu belirtilen sütun veya sütunların eşitliğine göre ortak sütunlar üzerinden birleştirir. Kullanımında JOIN ifadesinin yanı sıra INNER JOIN şeklinde de tercih edilebilir.

Örnek:

```
SELECT * FROM tbl_ogrenci INNER JOIN tbl_kulup ON tbl_ogrenci.kulup_no=tbl_kulup.kulup_no
```

Yukarıdaki sorgunun sonucunda, tbl_ogrenci ve tbl_kulup tablolarında eşleşen veriler getirilmektedir. Hiçbir kulüpte olmayan öğrenciler (Kağan) ile hiçbir öğrencinin olmadığı kulüpler (Robotik) burada gösterilmemektedir.

	ogrenci_no	ad	kulup_no	kulup_no	kulup_ad
1	1	İsmail	1	1	Siber Güvenlik
2	2	Hami	1	1	Siber Güvenlik
3	3	Meryem	2	2	Yapay Zeka
4	4	Durmuş	2	2	Yapay Zeka

OUTER JOIN: OUTER JOIN temelde INNER JOIN ile benzer şekilde ortak sütunlar üzerinden bağlantı kurar, ancak sonuç olarak tablolarda karşılığı olmayan satırları getirir. Bu sayede bir veri tablosunda olup bir diğerinde (veya diğerlerinde) olmayan verileri listeleyebiliriz. Kullanım aşamasında baskın olacak (temel olarak alınacak) tablo ayrıca LEFT ve RIGHT ifadeleriyle belirtilir ve belirtilen ikinci tabloda karşılığı olmayan satırlar *NULL* olarak gösterilir.

LEFT OUTER JOIN veya LEFT JOIN: Dış birleştiricilerden LEFT JOIN solunda belirtilen tablodaki tüm satırları listeler ve sağ tarafta belirtilen tabloya ait sütunları ise *NULL* olarak döndürür.

Örnek:

```
SELECT * FROM tbl_ogrenci LEFT JOIN tbl_kulup ON tbl_ogrenci.kulup_no=tbl_kulup.kulup_no
```

Yukarıdaki sorgunun sonucunda, Kağan isimli öğrencinin hiçbir kulüpte olmadığı görülmekteyken, diğer tüm öğrenciler ve üye oldukları kulüpler karşılarında yazmaktadır.

	ogrenci_no	ad	kulup_no	kulup_no	kulup_ad
1	1	İsmail	1	1	Siber Güvenlik
2	2	Hami	1	1	Siber Güvenlik
3	3	Meryem	2	2	Yapay Zeka
4	4	Durmuş	2	2	Yapay Zeka
5	5	Kağan	NULL	NULL	NULL

RIGHT OUTER JOIN veya RIGHT JOIN: RIGHT JOIN, LEFT JOIN gibi yön belirtrerek sağında yer alan tablodaki tüm satırları listeler. Solunda yer alan tablo NULL olarak dönmektedir.

Örnek:

```
SELECT * FROM tbl_ogrenci RIGHT JOIN tbl_kulup ON tbl_ogrenci.kulup_no=tbl_kulup.kulup_no
```

Yukarıdaki sorgu ile tüm kulüpleri ve bunlara üye olan öğrenciler gösterilmektedir. Dikkat edilirse Robotik kulübüne üye olan herhangi bir öğrenci bilgisi yoktur.

	ogrenci_no	ad	kulup_no	kulup_no	kulup_ad
1	1	İsmail	1	1	Siber Güvenlik
2	2	Hami	1	1	Siber Güvenlik
3	3	Meryem	2	2	Yapay Zeka
4	4	Durmuş	2	2	Yapay Zeka
5	NULL	NULL	NULL	3	Robotik

FULL JOIN veya FULL OUTER JOIN: FULL JOIN, arasında belirtildiği tablolara ait (sağ ve sol) tüm satırları listeler ve bu tablolarda karşılığı olmayan satırları *NULL* olarak belirtir. FULL JOIN işlemi LEFT ve RIGHT JOIN işlemlerinin UNION ile birleştirilmesi şeklinde de uygulanabilir.

Örnek:

```
SELECT * FROM tbl_ogrenci FULL OUTER JOIN tbl_kulup ON tbl_ogrenci.kulup_no=tbl_kulup.kulup_no
```

Yukarıdaki sorgu sonucunda, hem Kağan isimli öğrenciyi hem de Robotik kulübünü aynı anda görebilmekteyiz ancak bunlarla eşleşen kayıtlar olmadığı için karşılarında NULL ifadeleri yar almaktadır.

	ogrenci_no	ad	kulup_no	kulup_no	kulup_ad
1	1	İsmail	1	1	Siber Güvenlik
2	2	Hami	1	1	Siber Güvenlik
3	3	Meryem	2	2	Yapay Zeka
4	4	Durmuş	2	2	Yapay Zeka
5	5	Kağan	NULL	NULL	NULL
6	NULL	NULL	NULL	3	Robotik

CROSS JOIN: Tablolar arasında yapılan birleştirmede seçilen sütunlar arasındaki tüm kombinasyonlar sonuç tablosu (result table) haline getirilir. Bu sonuç tablosundaki satır sayısı alanların kartezyen çarpımı (cartesian product / iki kümenin elemanlarının sırayla karşılaştırılması) kadardır.

Örnek:

```
SELECT * FROM tbl_ogrenci CROSS JOIN tbl_kulup
```

	ogrenci_no	ad	kulup_no	kulup_no	kulup_ad
1	1	İsmail	1	3	Robotik
2	2	Hami	1	3	Robotik
3	3	Meryem	2	3	Robotik
4	4	Durmuş	2	3	Robotik
5	5	Kağan	NULL	3	Robotik
6	1	İsmail	1	1	Siber Güvenlik
7	2	Hami	1	1	Siber Güvenlik
8	3	Meryem	2	1	Siber Güvenlik
9	4	Durmuş	2	1	Siber Güvenlik
10	5	Kağan	NULL	1	Siber Güvenlik
11	1	İsmail	1	2	Yapay Zeka
12	2	Hami	1	2	Yapay Zeka
13	3	Meryem	2	2	Yapay Zeka
14	4	Durmuş	2	2	Yapay Zeka
15	5	Kağan	NULL	2	Yapay Zeka

SQL INSERT, UPDATE, DELETE KOMUTLARI

INSERT: Bir tabloya yeni veri eklemek için **INSERT** deyimi kullanılır. INSERT deyimi, INTO ve VALUES ifadeleri ile birlikte kullanılır ve tabloya yeni veri eklenmesi sağlanır. Veri ekleme sırasında ilk olarak INSERT INTO ile ekleme yapılacak olan tablo veya sütunlar belirlenir. Daha sonra VALUES ifadesi ile eklenecek olan değerler parantez içinde belirtilir. Verilerin sıralanışına dikkat etmek gerekmektedir. Örneğin (Personel_no, Adı, Soyadı, Bölümü) şeklinde sıralanmış olan alanlarda, eklenecek değerlerin de aynı sırada olması gerekmektedir. Tablodaki tüm alanlara değer girilecekse, tablo isminden sonra sütun isimlerinin belirtilmesine gerek yoktur. Çünkü bu tür durumda tablodaki tüm alanlara veri girişi sağlanmış olacaktır.

Sütun ismi belirtmeden kullanımı:

INSERT INTO tablo VALUES (değer1, değer2…)

Sütun ismi belirterek kullanımı:

INSERT INTO tablo (sutun1, sutun2…) **VALUES** (deger1, deger2…)

Örnek: Müşteriler tablosuna yeni müşteri eklemek için;

```
INSERT INTO musteriler (musteriNo, ad, soyad, dogumTarihi,sehir, cinsiyet, puan) VALUES ('11','Alexander','Sörloth','01.01.1994','Trabzon','E','90')
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
3	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20
4	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40
5	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84
6	6	11	Alexander	Sörloth	1994-01-01 00:00:00.000	Trabzon	E	90

UPDATE: Bir tabloda bulunan kayıt veya kayıtların istenildiği zaman değiştirilmesi mümkündür. Tablolarda güncelleme işlemini gerçekleştirmek için UPDATE komutu kullanılır. SET ifadesi ile güncellenecek alanlar ve bu alanların alacakları yeni değerler belirlenir. WHERE deyimi ile de verilerin güncelleştirilmesi için koşul belirlenir. Eğer WHERE ifadesi ile bir koşul belirlenmezse tablodaki tüm kayıtlar güncellenmiş olacağından WHERE kullanmaya dikkat edilmesi gerekmektedir.

Kullanımı:

UPDATE tablo **SET** sutun1=deger1, sutun2=deger2,… **WHERE** Koşul

Örnek: Müşteriler tablosunda Alexander Sörloth isimli müşterinin puan bilgisini tablodaki id numarasına göre güncellemek için;

```
UPDATE musteriler SET puan='85' WHERE id=6
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
3	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20
4	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40
5	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84
6	6	11	Alexander	Sörloth	1994-01-01 00:00:00.000	Trabzon	E	85

DELETE: Bir tabloda bulunan kayıt veya kayıtların istenildiği zaman silinmesi mümkündür. Tablolarda silme işlemini gerçekleştirmek için DELETE komutu kullanılır. DELETE komutu kullanılırken FROM eki ile birlikte tablo ismi yazılarak hangi tablodan veri silinmesi istendiği belirtilebilir. WHERE deyimi ile de verilerin silinme koşulu belirlenir. Eğer WHERE ifadesi ile bir koşul belirlenmezse tablodaki tüm kayıtlar silineceğinden WHERE kullanmaya dikkat edilmesi gerekmektedir.

Kullanımı:

DELETE FROM tablo **WHERE** koşul

Örnek: Müşteriler tablosunda Alexander Sörloth isimli müşteriyi id numarasına göre silmek için;

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
3	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20
4	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40
5	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84

ALTER KOMUTU

ALTER TABLE komutu ile bir tablonun yapısında değişiklik yapmak mümkündür.

Mevcut Bir Tabloya Sütun Ekleme

Tabloya alan(sütun) eklemek için ALTER TABLE komutuna ADD sözcüğü eklenmelidir.

Örnek: Müşteriler tablosuna "meslek" sütunu eklemek için;

```
ALTER TABLE musteriler ADD meslek NVARCHAR(30)
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan	meslek
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75	NULL
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61	NULL
3	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20	NULL
4	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40	NULL
5	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84	NULL

Mevcut Sütun Üzerinde Değişiklik Yapmak

Mevcut bir sütun üzerinde değişiklik yapma, değişken boyutlu bir veri tipine sahip olan sütunun genişliğini arttırma ile sınırlıdır. Bu anlamda, sütun genişliğini azaltma ya da veri tipini değiştirme mümkündür. Bu işlem için ALTER TABLE komutuna ALTER COLUMN deyiminin eklenmesi ile gerçekleştirilir.

Örnek: Müşteriler tablosundaki meslek sütununun değişken boyutunu ve veri tipini değiştirmek için;

```
ALTER TABLE musteriler ALTER COLUMN meslek CHAR(40)
```

Mevcut Bir Tablodan Sütun Silmek

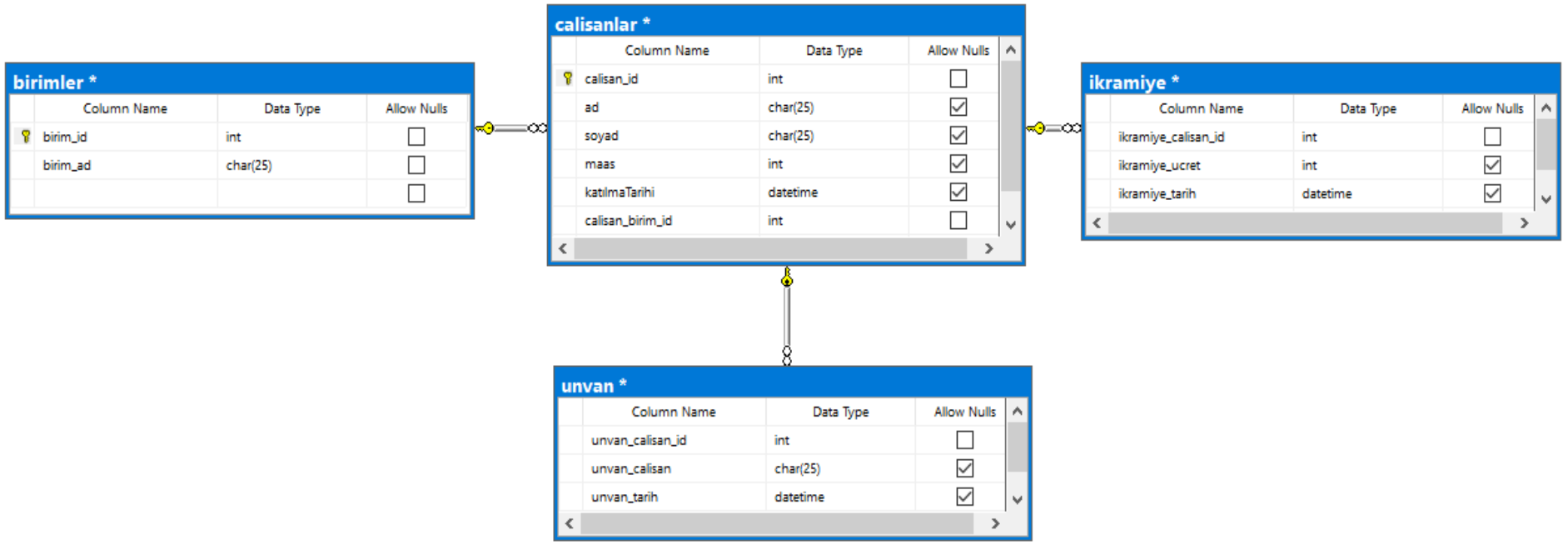
Tablo üzerinden sütun silmek için, ALTER TABLE komutuna DROP COLUMN deyiminin eklenmesi gerekmektedir.

Örnek: Müşteriler tablosundan meslek sütununu silmek için;

```
ALTER TABLE musteriler DROP COLUMN meslek
```

	id	musteriNo	ad	soyad	dogumTarihi	sehir	cinsiyet	puan
1	1	1	İsmail	İşeri	1981-01-01 00:00:00.000	Tokat	E	75
2	2	2	Hami	Satılmış	1992-01-01 00:00:00.000	Samsun	E	61
3	3	3	Kağan	Yazar	1993-01-01 00:00:00.000	Samsun	E	20
4	4	4	Meryem	Soysaldı	1990-01-01 00:00:00.000	Nevşehir	K	40
5	5	5	Durmuş	Şahin	1990-01-01 00:00:00.000	Mersin	E	84

SORULAR



1) Yukarıda bulunan 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 ilişkilerde önemli olduğunu unutmayınız. (Veritabanı ve tabloları SQL sorguları ile yapılmalıdır.)

2) Yukarıdaki diyagramın veritabanını oluşturduktan sonra tablolara SQL sorguları ile veri girişi yapınız. Girilecek örnek veriler ve veriler girildikten sonraki tabloların son hali aşağıdaki görselde bulunmaktadır.

Birim tablosu

	birim_id	birim_ad
1	1	Yazılım
2	2	Donanım
3	3	Güvenlik

Calisan tablosu

	calisan_id	ad	soyad	maas	katılmaTarihi	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	2014-02-20 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

İkramiye Tablosu

	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

Unvan Tablosu

	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

3) "Yazılım" veya "Donanım" birimlerinde çalışanların ad, soyad ve maaş bilgilerini listeleyen SQL sorgusunu yazınız. (Tek bir sorgu ile)

4) Maaşı en yüksek olan çalışanların ad, soyad ve maaş bilgilerini listeleyen SQL sorgusunu yazınız. (Tek bir sorgu ile)

5) Birimlerin her birinde kaç adet çalışan olduğunu ve birimlerin isimlerini listeleyen sorguyu yazınız. (Tek bir sorgu ile) (Örneğin; x biriminde 3 çalışan var gibi düşünebilirsiniz.) (Gruplamalı sorgu ile)

6) Birden fazla çalışana ait olan unvanların isimlerini ve o unvan altında çalışanların sayısını listeleyen sorguyu yazınız. (Tek bir sorgu ile)

7) Maaşları "50000" ve "100000" arasında değişen çalışanların ad, soyad ve maaş bilgilerini listeleyen sorguyu yazınız. (Tek bir sorgu ile)

8) İkramiye hakkına sahip çalışanlara ait ad, soyad, birim, unvan ve ikramiye ücreti bilgilerini listeleyen sorguyu yazınız. (Tek bir sorgu ile)

9) Ünvanı "Yönetici" ve "Müdür" olan çalışanların ad, soyad ve ünvanlarını listeleyen sorguyu yazınız. (Tek bir sorgu ile)

10) Her bir birimde en yüksek maaş alan çalışanların ad, soyad ve maaş bilgilerini listeleyen sorguyu yazınız. (Tek bir sorgu ile)

NOT:

- **1. ve 2. sorular SQL sorguları ile yapılacaktır. (SQL Management Studio'nun sunmuş olduğu tablo oluşturma ve tablolara veri ekleme araçları ile yapanlar bu sorulardan 0 puan alacaklardır.)**
- **Diğer sorularda listelenmesi istenilen veriler tek bir sorgu cümlesi ile listelenmelidirler. (İç içe sorgular da tek bir sorguya dahildir.)**

- **Soruların çözümleri için yazdığınız SQL sorguları ve sorguların çıktıları ekran görüntüsü olarak raporunuzda yer almalıdır.**
- **Laboratuvara gelmeyenlerin (raporlarını teslim etseler bile) föy notlarına 0 verilecektir.**
- **Laboratuvara gelip, raporlarını teslim etmeyenlerin de föy notlarına 0 verilecektir.**
- **Raporlarınız da değerlendirileceği için raporlarınıza özen göstermeniz gerekmektedir.**
- **Raporlarınızı size belirtilen formatta ve zamanında yükleyiniz. Raporunuzda, sizden yapılması istenen sorguların bulunduğu GitHub linkinin olması gerekmektedir. GitHub linki bulunmayan raporlar değerlendirilmeyecektir ve föy notu 0 olarak değerlendirilecektir.**