Database Bootcamp – Ödev 1

Bu ödevde, verilen SQL sorularına uygun sorgular yazıldı ve her birinin açıklaması detaylandırıldı.

1. Soru: Çalışanların sadece FirstName, LastName ve Salary bilgilerini getiren bir SQL sorgusu yazınız.

SQL Sorgusu:

SELECT firstname, lastname, salary

FROM employees;

```
--A seçeneği için çözümler;

SELECT firstname, lastname, salary FROM employees
--SELECT firstname FROM employees
--SELECT lastname FROM employees
--SELECT salary FROM employees
```

Data Output Messages Notifications

	firstname character varying (50)	lastname character varying (50)	salary numeric (10,2)
1	John	Doe	55000.00
2	Jane	Smith	65000.00
3	Sam	Brown	52000.00
4	Lisa	White	70000.00
5	Mark	Black	75000.00
6	Lucy	Green	60000.00

Bu sorguda, employees tablosundan **SELECT** sorgusu ile firstname, lastname ve salary sütunlarını seçtim.

FROM ifadesi, bu verilerin hangi tablodan çekileceğini belirtir. Ayrıca, ihtiyaca göre belirli sütunları ayrı ayrı sorgulayarak da veri çekebiliriz.

SELECT firstname FROM employees

SELECT lastname FROM employees

SELECT salary FROM employees

Gibi sorgularla yalnızca istenen sütun da tablodan çekilebilir.

2. Soru: Çalışanların çalıştıkları departmanları benzersiz olarak listeleyen bir SQL sorgusu yazınız.

SQL Sorgusu:

SELECT DISTINCT d.departmentname

FROM employees e

INNER JOIN departments d ON e.departmentid = d.departmentid;

```
8   --B seçeneği için çözüm;
9
10  SELECT DISTINCT d.departmentname
11  FROM employees e
12  INNER JOIN departments d ON e.departmentid = d.departmentid;
```

Data Output | Messages | Notifications

	departmentname character varying (50)
1	Finance
2	IT
3	HR

Bu sorguda, employees ve departments tablolarını **INNER JOIN** ile departmentid sütunu üzerinden birleştirdim. **INNER JOIN** tercih etmemin sebebi, yalnızca eşleşen satırları getirmek istemem ve bu işlemin **JOIN** işlemine kıyasla daha düşük maliyetli olmasıdır.

Daha sonra, çalışanların bağlı olduğu departmentname sütununu sorguya dahil ettim. **DISTINCT** ifadesi, tekrar eden departman isimlerini engelleyerek her departmanı yalnızca bir kez listelememi sağladı.

```
8   --B seçeneği için çözüm;
9
10  SELECT d.departmentname
11  FROM employees e
12  INNER JOIN departments d ON e.departmentid = d.departmentid;
```

Data Output | Messages | Notifications

	departmentname character varying (50)
1	IT
2	HR
3	IT
4	Finance
5	IT
6	HR

Böylece, sistemde kayıtlı ve en az bir çalışanı bulunan benzersiz departmanların listesini elde ettim. Eğer **DISTINCT** kullanmasaydım, employees tablosunda yer alan tüm çalışanların departmanları tekrar edecek şekilde listelenecekti.

3. Soru: Sadece IT departmanında çalışanların bilgilerini getiren bir SQL sorgusu yazınız.

SQL Sorgusu:

SELECT * **FROM** employees e

INNER JOIN departments d **ON** e.departmentid = d.departmentid

WHERE d.departmentname = **'IT'**

```
14  --C seçeneği için çözüm;
15
16  SELECT * FROM employees e
17  INNER JOIN departments d ON e.departmentid = d.departmentid
18  WHERE d.departmentname = 'IT'
```

Data Output Messages Notifications

Showing ro

	employeeid integer	firstname character varying (50)	lastname character varying (50)	age integer	departmentid integer	salary numeric (10,2)	joindate date	departmentid integer	departmentname character varying (50)
1	1	John	Doe	30	1	55000.00	2020-01-15	1	IT
2	3	Sam	Brown	28	1	52000.00	2021-04-25	1	IT
3	5	Mark	Black	50	1	75000.00	2015-11-05	1	IT

Bu sorguda, employees tablosundaki tüm sütunları seçtim ve **INNER JOIN** kullanarak departments tablosuyla departmentid sütunu üzerinden birleştirdim.

WHERE koşulunda yalnızca departmentname değeri **"IT"** olan satırların getirilmesini sağladım.

Ayrıca, tablo adlarını **employees e** ve **departments d** olarak kısaltarak kullandım. Bu sayede, departments.departmentname yerine d.departmentname gibi daha sade bir yazım kullanarak sorguyu daha okunaklı ve kullanışlı hale getirdim.

```
16  SELECT e.firstname, e.lastname, d.departmentname
17  FROM employees e
18  INNER JOIN departments d ON e.departmentid = d.departmentid
19  WHERE d.departmentname = 'IT'
```

Data Output Messages Notifications

	firstname character varying (50)	lastname character varying (50)	departmentname character varying (50)
1	John	Doe	IT
2	Sam	Brown	IT
3	Mark	Black	IT

Burada ise yalnızca **IT** departmanında çalışan kişilerin adını ve soyadını tabloda görmek istedim, böylece sorgunun daha sade ve anlaşılır olmasını sağladım.

Bunu sağlamak için, önceki örnekte olduğu gibi **SELECT** sorgusunu kullandım ve yalnızca ihtiyacım olan sütunları seçtim.

4. Soru: Çalışanları maaşlarına göre büyükten küçüğe sıralayan bir SQL sorgusu yazınız.

SQL Sorgusu:

```
SELECT e.firstname, e.lastname, e.salary

FROM employees e

ORDER BY salary DESC
```

```
26  --D seçeneği için çözüm
27
28  SELECT e.firstname, e.lastname, e.salary
29  FROM employees e
30  ORDER BY salary DESC
31
```

Data Output Messages Notifications

	firstname character varying (50)	lastname character varying (50)	salary numeric (10,2)
1	Mark	Black	75000.00
2	Lisa	White	70000.00
3	Jane	Smith	65000.00
4	Lucy	Green	60000.00
5	John	Doe	55000.00
6	Sam	Brown	52000.00

Bu sorguda, employees tablosundan firstname, lastname ve salary sütunlarını seçtim. **ORDER BY** ifadesi, sorgu sonucundaki verileri belirli bir sütuna göre sıralamak için kullanılır.

Bu örnekte, sıralamanın salary sütununa göre yapılması gerektiği için **ORDER** BY salary ifadesini kullandım. Maaşların büyükten küçüğe sıralanmasını sağlamak için **DESC (Azalan sıralama)** ifadesini ekledim.

5. Soru: Çalışanların FirstName ve LastName alanlarını birleştirerek, tam adlarını içeren yeni bir sütun oluşturan bir SQL sorgusu yazınız.

SQL Sorgusu:

SELECT e.firstname **|| ' ' ||** e.lastname **AS** full_name

FROM employees e**;**

```
--E seçeneği için çözüm;

SELECT e.firstname || ' ' || e.lastname AS full_name
FROM employees e;
```

Data Output Messages Notifications

	full_name text
1	John Doe
2	Jane Smith
3	Sam Brown
4	Lisa White
5	Mark Black
6	Lucy Green

Bu sorguda, çalışanların **firstname** ve **lastname** sütunlarını birleştirerek tam isimlerini içeren **full_name** adında yeni bir sütun oluşturdum. **CONCAT** fonksiyonu, MySQL gibi veritabanlarında metinleri birleştirmek için kullanılır. **PostgreSQL** gibi sistemlerde || ' ' || operatörü kullanılsa da **Tuncay hocamız**, MySQL'de **CONCAT** fonksiyonunun kullanılması gerektiğini belirtti.

Ayrıca, ' ' ifadesini ekleyerek ad ve soyad arasında bir boşluk olmasını sağladım. **AS full_name** ifadesiyle de oluşturduğum yeni sütuna anlamlı bir isim verdim.

Bu ödevde, farklı SQL sorgularını kullanarak belirlenen kriterlere uygun veri çekme işlemleri gerçekleştirildi. **SQL'in temel sorgulama yapıları** olan **SELECT, JOIN, WHERE, DISTINCT ve ORDER BY** gibi ifadeler pratik edildi.