7.Bölüm Cevaplar

İnceleme Soruları ve Cevapları:

1. **Tarih verilerini saklamak için karakter veri türü yerine DATE veri türünü kullanmanın neden tercih edileceğini açıklayın.**

   Cevap: DATE veri türü, tarihsel veriler için optimize edilmiştir. Tarihlerle aritmetik işlemler yapmayı, sıralamayı ve karşılaştırmayı kolaylaştırır. Karakter veri türü ile saklanan tarih bilgileri üzerinde doğrudan matematiksel işlemler yapılamaz ve formatlama hatalarına açıktır.

2. **Aşağıdaki komutun neden bir hata oluşturduğunu ve düzeltmek için ne gibi değişiklikler yapılabileceğini açıklayın:**

   ```
   SEÇİNİZ V_CODE, SUM(P_QOH)
   ÜRÜNDEN;
   ```

   Cevap:

   - Hata nedeni: GROUP BY ifadesinin eksik olmasıdır. SUM gibi toplama fonksiyonları kullanılırken, gruplanmamış sütunlar için GROUP BY kullanmak gerekir.
   - Düzeltme:

     ```
     SELECT V_CODE, SUM(P_QOH)
     FROM ÜRÜN
     GROUP BY V_CODE;
     ```

3. **Çapraz birleştirme (CROSS JOIN) nedir? Sözdizimine bir örnek verin.**

   Cevap: Çapraz birleştirme, iki tabloyu koşulsuz olarak birleştirerek her satırı diğer tablodaki tüm satırlarla eşleştirir.

   Örnek:

   ```
   SELECT * FROM T1
   ```

```
CROSS JOIN T2;
```

4. **Dış birleştirme sınıflandırmasına hangi üç birleştirme türü dahildir?**

   Cevap:

   - LEFT OUTER JOIN
   - RIGHT OUTER JOIN
   - FULL OUTER JOIN

5. **Üç dış birleştirme türü için birer sorgu yazın.**

```
-- LEFT OUTER JOIN
SELECT * FROM T1
LEFT JOIN T2 ON T1.C1 = T2.C1;

-- RIGHT OUTER JOIN
SELECT * FROM T1
RIGHT JOIN T2 ON T1.C1 = T2.C1;

-- FULL OUTER JOIN
SELECT * FROM T1
FULL JOIN T2 ON T1.C1 = T2.C1;
```

6. **Özyinelemeli (Recursive) nedir?**

   Cevap: Özyinelemeli, bir işlemin kendisini çağırarak çalışmasını ifade eder. SQL'de özyinelemeli sorgular genellikle hiyerarşik veri yapıları (örneğin, çalışanların yöneticileriyle ilişkilendirilmesi) için kullanılır.

7. **WHERE cümlesini IN operatörü olmadan yazın.**

```
WHERE V_STATE = 'TN' OR V_STATE = 'FL' OR V_STATE = 'GA'
```

8. **ORDER BY ve GROUP BY arasındaki fark?**

   Cevap:

   - **ORDER BY:** Sonuç kümesini belirtilen sütunlara göre sıralar.

- **GROUP BY:** Verileri belirli sütunlara göre gruplar ve her grup için toplama işlemleri yapılmasını sağlar.

9. **Aşağıdaki iki sorgu neden farklı sonuçlar üretir?**

```
SELECT DISTINCT COUNT(V_CODE) FROM PRODUCT;
SELECT COUNT(DISTINCT V_CODE) FROM PRODUCT;
```

Cevap:

- İlk sorgu, COUNT(V_CODE) değerini hesapladıktan sonra DISTINCT uygular.
- İkinci sorgu, V_CODE sütunundaki benzersiz değerleri sayar.

10. **COUNT ve SUM işlevleri arasındaki fark?**

Cevap:

- **COUNT:** Satır sayısını döndürür.
- **SUM:** Sayısal sütunların toplamını döndürür.

11. **WHERE ve HAVING arasındaki fark?**

Cevap:

- **WHERE:** Satırları filtreler, gruplama öncesi kullanılır.
- **HAVING:** Gruplanmış sonuçları filtreler.

12. **Alt sorgu nedir?**

Cevap: Ana sorgu içinde kullanılan, sonuç döndüren iç sorgudur.

13. **Alt sorguların döndürebileceği üç sonuç türü?**

Cevap:

- Tek değer (Scalar)
- Tek sütunlu liste (Column)
- Çok satır ve sütun içeren sonuç kümesi (Table)

14. **İlişkili alt sorgu nedir?**

Cevap: Dış sorgudaki her satır için çalıştırılan alt sorgudur.

Örnek:

```
SELECT * FROM EMPLOYEE E
WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE WHERE DEP
T_ID = E.DEPT_ID);
```

15. **Normal alt sorgu ile ilişkili alt sorgu arasındaki fark?**

    Cevap: Normal alt sorgular bağımsızdır, ilişkili alt sorgular dış sorgunun her satırına bağlıdır.

16. **SQL operatörleri küme yönelimli midir?**

    Cevap: Evet, çünkü SQL, birden fazla satır ve sütunu aynı anda işler.

17. **Union uyumluluk (union compatible) nedir?**

    Cevap: UNION, INTERSECT, EXCEPT işlemlerinin çalışabilmesi için sütun sayısı ve veri türleri aynı olmalıdır.

18. **UNION ve UNION ALL arasındaki fark?**

    Cevap:

    - **UNION:** Tekrar eden satırları kaldırır.
    - **UNION ALL:** Tüm satırları gösterir.**Örnek:**

```
SELECT * FROM EMPLOYEE
UNION
SELECT * FROM EMPLOYEE_1;
```

19. **UNION çıktısı?**

    Cevap: Çalışanların birleşik kümesi, tekrar eden kayıtlar kaldırılmış şekilde listelenir.

20. **UNION ALL çıktısı?**

    Cevap: Tüm çalışanlar listelenir, tekrar eden kayıtlar dahil edilir.

21. **INTERSECT çıktısı?**

    Cevap: Ortak çalışanlar listelenir.

22. **EXCEPT (MINUS) çıktısı?**

    Cevap: EMPLOYEE'de olup EMPLOYEE_1'de olmayan çalışanlar listelenir.

23. **Verilen tablolar için UNION, UNION ALL, INTERSECT, EXCEPT çıktıları nedir?**

    Cevap:

    - **UNION:** Her iki tablodaki benzersiz VEND_CODE'ler
    - **UNION ALL:** Tüm VEND_CODE'ler
    - **INTERSECT:** Ortak VEND_CODE'ler
    - **EXCEPT:** PRODUCT tablosunda olup VENDOR'da olmayan VEND_CODE'ler

24. **EXCEPT (MINUS) sırası neden önemlidir?**

    Cevap: İlk tablodaki kayıtlar ikinci tablodakilerden çıkarılır. UNION'da sıralama fark etmez.

25. **MS Access ve SQL Server'da gün farkını hesaplayan fonksiyon?**

```
DATEDIFF(day, DoğumTarihi, GETDATE())
```

26. **Oracle'da gün farkını hesaplayan fonksiyon?**

```
SELECT SYSDATE - DoğumTarihi FROM DUAL;
```

27. **İlk üç karakteri getiren fonksiyon?**

    - **Oracle:**

```
SELECT SUBSTR(EMP_LNAME, 1, 3) FROM EMPLOYEE;
```

    - **SQL Server:**

```
SELECT LEFT(EMP_LNAME, 3) FROM EMPLOYEE;
```

28. **Bir SQL programcısı SELECT sorgusu oluşturmadan önce hangi iki şeyi anlamalıdır?**

    Cevap:

- Veritabanı şeması (tablolar ve ilişkiler)
- İstenen çıktının yapısı ve amacı

Problem Soruları ve Çözümleri:

1. Soyadı Smith ile başlayan tüm çalışanları listeleyin.

```
SELECT EMP_NUM, EMP_LNAME, EMP_FNAME, EMP_MNAME
FROM EMPLOYEE
WHERE EMP_LNAME LIKE 'Smith%'
ORDER BY EMP_NUM;
```

2. EMPLOYEE ve PROJECT tablolarını birleştirerek listeleyin.

```
SELECT E.EMP_NUM, E.EMP_LNAME, E.EMP_FNAME, P.PROJ_NAME
FROM EMPLOYEE E
JOIN ASSIGNMENT A ON E.EMP_NUM = A.EMP_NUM
JOIN PROJECT P ON A.PROJ_NUM = P.PROJ_NUM
ORDER BY P.PROJ_NAME;
```

3. Problem 2'deki sorguyu çalışanın soyadına göre sıralayın.

```
SELECT E.EMP_NUM, E.EMP_LNAME, E.EMP_FNAME, P.PROJ_NAME
FROM EMPLOYEE E
JOIN ASSIGNMENT A ON E.EMP_NUM = A.EMP_NUM
JOIN PROJECT P ON A.PROJ_NUM = P.PROJ_NUM
ORDER BY E.EMP_LNAME;
```

4. ASSIGNMENT tablosunda farklı proje numaralarını listeleyin.

```
SELECT DISTINCT PROJ_NUM
FROM ASSIGNMENT
ORDER BY PROJ_NUM;
```

5. ASSIGNMENT tablosundaki ücretleri doğrulayan bir sorgu yazın.

```
SELECT ASSIGN_NUM, EMP_NUM, PROJ_NUM,
      ASSIGN_CHARGE, (ASSIGN_CHG_HR * ASSIGN_HOURS) AS CALCULATE
D_CHARGE
FROM ASSIGNMENT
ORDER BY ASSIGN_NUM;
```

6. Çalışılan toplam saatleri ve ücretleri listeleyin.

```
SELECT EMP_NUM, SUM(ASSIGN_HOURS) AS TOTAL_HOURS,
      SUM(ASSIGN_CHG_HR * ASSIGN_HOURS) AS TOTAL_CHARGE
FROM ASSIGNMENT
GROUP BY EMP_NUM
ORDER BY EMP_NUM;
```

7. Proje bazında toplam saat ve ücretleri listeleyin.

```
SELECT PROJ_NUM, SUM(ASSIGN_HOURS) AS TOTAL_HOURS,
      SUM(ASSIGN_CHG_HR * ASSIGN_HOURS) AS TOTAL_CHARGE
FROM ASSIGNMENT
GROUP BY PROJ_NUM
ORDER BY PROJ_NUM;
```

8. Tüm çalışanlar için toplam saat ve masrafları hesaplayın.

```
SELECT SUM(ASSIGN_HOURS) AS TOTAL_HOURS,
      SUM(ASSIGN_CHG_HR * ASSIGN_HOURS) AS TOTAL_COST
FROM ASSIGNMENT;
```

9. Fatura sayısını sayın.

```
SELECT COUNT(*) AS INVOICE_COUNT
FROM INVOICE;
```

10. Bakiyesi 500$'dan fazla olan müşteri sayısını bulun.

```
SELECT COUNT(*) AS HIGH_BALANCE_CUSTOMERS
FROM CUSTOMER
WHERE BALANCE > 500;
```

11. Müşterilerin yaptığı tüm satın alımları listeleyin.

```
SELECT C.CUST_CODE, I.INV_NUM, P.PROD_DESCRIPT
FROM CUSTOMER C
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
JOIN LINE L ON I.INV_NUM = L.INV_NUM
JOIN PRODUCT P ON L.PROD_SKU = P.PROD_SKU
ORDER BY C.CUST_CODE, I.INV_NUM, P.PROD_DESCRIPT;
```

12. Müşteri satın alımları için alt toplamları hesaplayın.

```
SELECT C.CUST_CODE, I.INV_NUM, P.PROD_DESCRIPT,
    (L.LINE_UNITS * L.LINE_PRICE) AS SUBTOTAL
FROM CUSTOMER C
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
JOIN LINE L ON I.INV_NUM = L.INV_NUM
JOIN PRODUCT P ON L.PROD_SKU = P.PROD_SKU
ORDER BY C.CUST_CODE, I.INV_NUM, P.PROD_DESCRIPT;
```

13. Her müşteri için toplam satın alma değerini bulun.

```
SELECT C.CUST_CODE, C.BALANCE,
    SUM(L.LINE_UNITS * L.LINE_PRICE) AS TOTAL_PURCHASE
FROM CUSTOMER C
```

```
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
JOIN LINE L ON I.INV_NUM = L.INV_NUM
GROUP BY C.CUST_CODE, C.BALANCE
ORDER BY C.CUST_CODE;
```

14. Müşteri satın alımlarının sayısını listeleyin.

```
SELECT C.CUST_CODE, COUNT(L.LINE_UNITS) AS TOTAL_PURCHASES
FROM CUSTOMER C
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
JOIN LINE L ON I.INV_NUM = L.INV_NUM
GROUP BY C.CUST_CODE
ORDER BY C.CUST_CODE;
```

15. Müşterilerin ortalama satın alma tutarlarını bulun.

```
SELECT C.CUST_CODE, SUM(L.LINE_UNITS * L.LINE_PRICE) AS TOTAL_PURC
HASE,
    COUNT(L.LINE_UNITS) AS TOTAL_ORDERS,
    AVG(L.LINE_UNITS * L.LINE_PRICE) AS AVG_PURCHASE
FROM CUSTOMER C
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
JOIN LINE L ON I.INV_NUM = L.INV_NUM
GROUP BY C.CUST_CODE
ORDER BY C.CUST_CODE;
```

16. Her fatura için toplam satın alma değerini hesaplayın.

```
SELECT I.INV_NUM, SUM(L.LINE_UNITS * L.LINE_PRICE) AS TOTAL_INVOICE_
AMOUNT
FROM INVOICE I
JOIN LINE L ON I.INV_NUM = L.INV_NUM
```

```
GROUP BY I.INV_NUM
ORDER BY I.INV_NUM;
```

17. Müşteriye göre fatura sayısı ve toplam satın alma tutarlarını üretin.

```
SELECT C.CUST_CODE, COUNT(I.INV_NUM) AS TOTAL_INVOICES,
      SUM(L.LINE_UNITS * L.LINE_PRICE) AS TOTAL_PURCHASE
FROM CUSTOMER C
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
JOIN LINE L ON I.INV_NUM = L.INV_NUM
GROUP BY C.CUST_CODE
ORDER BY C.CUST_CODE;
```

18. Tüm faturaların toplamı, en küçük, en büyük ve ortalama satışları hesaplayın.

```
SELECT COUNT(I.INV_NUM) AS TOTAL_INVOICES,
      SUM(L.LINE_UNITS * L.LINE_PRICE) AS TOTAL_SALES,
      MIN(L.LINE_UNITS * L.LINE_PRICE) AS MIN_PURCHASE,
      MAX(L.LINE_UNITS * L.LINE_PRICE) AS MAX_PURCHASE,
      AVG(L.LINE_UNITS * L.LINE_PRICE) AS AVG_PURCHASE
FROM INVOICE I
JOIN LINE L ON I.INV_NUM = L.INV_NUM;
```

19. Faturalama döneminde satın alma yapmış müşterilerin bakiyelerini listeleyin.

```
SELECT DISTINCT C.CUST_CODE, C.BALANCE
FROM CUSTOMER C
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
ORDER BY C.CUST_CODE;
```

20. Satın alma yapmayan müşterileri listeleyin.

```
SELECT C.CUST_CODE, C.BALANCE
FROM CUSTOMER C
LEFT JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
WHERE I.INV_NUM IS NULL
ORDER BY C.CUST_CODE;
```

21. Envanterdeki tüm ürünlerin toplam değerini hesaplayın.

```
SELECT SUM(PROD_QOH * PROD_PRICE) AS TOTAL_INVENTORY_VALUE
FROM PRODUCT;
```

22. Departman 300'de çalışan veya "CLERK I" unvanına sahip çalışanları listeleyin.

```
SELECT EMP_NUM, EMP_FNAME, EMP_LNAME, EMP_PHONE, EMP_TITLE, DE
PT_NUM
FROM EMPLOYEE
WHERE DEPT_NUM = 300 OR EMP_TITLE = 'CLERK I'
ORDER BY EMP_LNAME, EMP_FNAME;
```

23. Belirli çalışanların maaş geçmişini görüntüleyin.

```
SELECT EMP_NUM, EMP_LNAME, SAL_FROM, SAL_END, SAL_AMOUNT
FROM EMPLOYEE E
JOIN SALARY_HISTORY S ON E.EMP_NUM = S.EMP_NUM
WHERE E.EMP_NUM IN (83731, 83745, 84039)
ORDER BY E.EMP_NUM, SAL_FROM;
```

24. Satıcıların sattığı marka ve ürün sayısını görüntüleyin.

```
SELECT V.VEND_ID, V.VEND_NAME, B.BRAND_NAME, COUNT(P.PROD_SKU)
AS TOTAL_PRODUCTS
FROM VENDOR V
```

```
JOIN SUPPLIES S ON V.VEND_ID = S.VEND_ID
JOIN PRODUCT P ON S.PROD_SKU = P.PROD_SKU
JOIN BRAND B ON P.BRAND_ID = B.BRAND_ID
GROUP BY V.VEND_ID, V.VEND_NAME, B.BRAND_NAME
ORDER BY V.VEND_NAME, B.BRAND_NAME;
```

25. En pahalı ürünü ve fiyatını görüntüleyin.

```
SELECT PROD_SKU, PROD_DESCRIPT, PROD_PRICE
FROM PRODUCT
ORDER BY PROD_PRICE DESC
LIMIT 1;
```

26. Bir müşteri için toplam satın alım sayısını listeleyin.

```
SELECT C.CUST_CODE, COUNT(DISTINCT I.INV_NUM) AS TOTAL_ORDERS
FROM CUSTOMER C
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
GROUP BY C.CUST_CODE
ORDER BY C.CUST_CODE;
```

27. Her markanın ürünlerinin ortalama fiyatını listeleyin.

```
SELECT B.BRAND_NAME, ROUND(AVG(P.PROD_PRICE), 2) AS AVG_PRICE
FROM BRAND B
JOIN PRODUCT P ON B.BRAND_ID = P.BRAND_ID
GROUP BY B.BRAND_NAME
ORDER BY B.BRAND_NAME;
```

28. Her departmandaki çalışan sayısını görüntüleyin.

```
SELECT DEPT_NUM, COUNT(EMP_NUM) AS TOTAL_EMPLOYEES
FROM EMPLOYEE
```

```
GROUP BY DEPT_NUM
ORDER BY DEPT_NUM;
```

29. 1 Mayıs 2011 ile 31 Aralık 2012 arasında işe alınan çalışanları listeleyin.

```
SELECT EMP_FNAME, EMP_LNAME, EMP_EMAIL
FROM EMPLOYEE
WHERE EMP_HIREDATE BETWEEN '2011-05-01' AND '2012-12-31'
ORDER BY EMP_LNAME, EMP_FNAME;
```

30. Bir departmanda çalışan en yüksek maaşlı çalışanları listeleyin.

```
SELECT EMP_NUM, EMP_FNAME, EMP_LNAME, MAX(SAL_AMOUNT) AS HIG
HEST_SALARY
FROM EMPLOYEE E
JOIN SALARY_HISTORY S ON E.EMP_NUM = S.EMP_NUM
WHERE DEPT_NUM = 200
GROUP BY EMP_NUM, EMP_FNAME, EMP_LNAME
ORDER BY HIGHEST_SALARY DESC;
```

31. Bir kitap ödünç alan farklı kullanıcıların sayısını görüntüleyin.

```
SELECT COUNT(DISTINCT PATRON_ID) AS TOTAL_BORROWERS
FROM CHECKOUT;
```

32. Fiyatı premium markalardan daha pahalı olan premium olmayan ürünleri listeleyin.

```
SELECT P.PROD_SKU, P.PROD_DESCRIPT, P.PROD_PRICE, B.BRAND_TYPE
FROM PRODUCT P
JOIN BRAND B ON P.BRAND_ID = B.BRAND_ID
WHERE B.BRAND_TYPE != 'Premium'
```

```
AND P.PROD_PRICE > (SELECT MAX(PROD_PRICE) FROM PRODUCT WHERE
BRAND_ID IN
                (SELECT BRAND_ID FROM BRAND WHERE BRAND_TYPE = 'Premi
um'))
ORDER BY P.PROD_PRICE DESC;
```

33. İş unvanı "ASSOCIATE" ile biten çalışanları listeleyin.

```
SELECT EMP_NUM, EMP_LNAME, EMP_EMAIL, EMP_TITLE, DEPT_NAME
FROM EMPLOYEE E
JOIN DEPARTMENT D ON E.DEPT_NUM = D.DEPT_NUM
WHERE EMP_TITLE LIKE '%ASSOCIATE'
ORDER BY DEPT_NAME, EMP_TITLE;
```

34. Her markanın veritabanında bulunan ürün sayısını listeleyin.

```
SELECT B.BRAND_NAME, COUNT(P.PROD_SKU) AS PRODUCT_COUNT
FROM BRAND B
JOIN PRODUCT P ON B.BRAND_ID = P.BRAND_ID
GROUP BY B.BRAND_NAME
ORDER BY B.BRAND_NAME;
```

35. Her kategorideki su bazlı ürünlerin sayısını görüntüleyin.

```
SELECT PROD_CATEGORY, COUNT(PROD_SKU) AS PRODUCT_COUNT
FROM PRODUCT
WHERE PROD_BASE = 'Su'
GROUP BY PROD_CATEGORY
ORDER BY PROD_CATEGORY;
```

36. Her baz ve tür kombinasyonundaki ürün sayılarını listeleyin.

```
SELECT PROD_BASE, PROD_TYPE, COUNT(PROD_SKU) AS PRODUCT_COUN
T
```

```
FROM PRODUCT
GROUP BY PROD_BASE, PROD_TYPE
ORDER BY PROD_BASE, PROD_TYPE;
```

37. Her markaya ait toplam envanteri görüntüleyin.

```
SELECT B.BRAND_NAME, SUM(P.PROD_QOH) AS TOTAL_INVENTORY
FROM BRAND B
JOIN PRODUCT P ON B.BRAND_ID = P.BRAND_ID
GROUP BY B.BRAND_NAME
ORDER BY TOTAL_INVENTORY DESC;
```

38. Her markanın ürünlerinin ortalama fiyatını listeleyin.

```
SELECT B.BRAND_NAME, ROUND(AVG(P.PROD_PRICE), 2) AS AVG_PRICE
FROM BRAND B
JOIN PRODUCT P ON B.BRAND_ID = P.BRAND_ID
GROUP BY B.BRAND_NAME
ORDER BY B.BRAND_NAME;
```

39. Her departmandaki en son işe alınan kişiyi görüntüleyin.

```
SELECT DEPT_NUM, MAX(EMP_HIREDATE) AS LAST_HIRED
FROM EMPLOYEE
GROUP BY DEPT_NUM
ORDER BY DEPT_NUM;
```

40. Departman 200'de çalışanların en yüksek maaşlarını listeleyin.

```
SELECT EMP_NUM, EMP_FNAME, EMP_LNAME, MAX(SAL_AMOUNT) AS HIG
HEST_SALARY
FROM EMPLOYEE E
JOIN SALARY_HISTORY S ON E.EMP_NUM = S.EMP_NUM
```

```
WHERE DEPT_NUM = 200
GROUP BY EMP_NUM, EMP_FNAME, EMP_LNAME
ORDER BY HIGHEST_SALARY DESC;
```

41. Kümülatif fatura toplamı 1.500$'dan büyük olan müşterileri listeleyin.

```
SELECT C.CUST_CODE, C.CUST_FNAME, C.CUST_LNAME,
      SUM(L.LINE_UNITS * L.LINE_PRICE) AS TOTAL_SPENT
FROM CUSTOMER C
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
JOIN LINE L ON I.INV_NUM = L.INV_NUM
GROUP BY C.CUST_CODE, C.CUST_FNAME, C.CUST_LNAME
HAVING TOTAL_SPENT > 1500
ORDER BY TOTAL_SPENT DESC;
```

42. Her departman yöneticisinin bilgilerini listeleyin.

```
SELECT D.DEPT_NUM, D.DEPT_NAME, D.DEPT_PHONE,
      E.EMP_NUM, E.EMP_LNAME AS MANAGER_LNAME
FROM DEPARTMENT D
JOIN EMPLOYEE E ON D.MANAGER_ID = E.EMP_NUM
ORDER BY D.DEPT_NAME;
```

43. Satıcıların tedarik ettiği marka ve ürün sayılarını listeleyin.

```
SELECT V.VEND_ID, V.VEND_NAME, B.BRAND_NAME, COUNT(P.PROD_SKU)
AS PRODUCT_COUNT
FROM VENDOR V
JOIN SUPPLIES S ON V.VEND_ID = S.VEND_ID
JOIN PRODUCT P ON S.PROD_SKU = P.PROD_SKU
JOIN BRAND B ON P.BRAND_ID = B.BRAND_ID
```

```
GROUP BY V.VEND_ID, V.VEND_NAME, B.BRAND_NAME
ORDER BY V.VEND_NAME, B.BRAND_NAME;
```

44. Her çalışan tarafından tamamlanan faturaların toplam değerini listeleyin.

```
SELECT E.EMP_NUM, E.EMP_LNAME, E.EMP_FNAME, SUM(I.INV_TOTAL) AS
TOTAL_SALES
FROM EMPLOYEE E
JOIN INVOICE I ON E.EMP_NUM = I.EMPLOYEE_ID
GROUP BY E.EMP_NUM, E.EMP_LNAME, E.EMP_FNAME
ORDER BY TOTAL_SALES DESC;
```

45. En büyük ortalama ürün fiyatına sahip markayı bulun.

```
SELECT B.BRAND_NAME, ROUND(AVG(P.PROD_PRICE), 2) AS AVG_PRICE
FROM BRAND B
JOIN PRODUCT P ON B.BRAND_ID = P.BRAND_ID
GROUP BY B.BRAND_NAME
ORDER BY AVG_PRICE DESC
LIMIT 1;
```

46. En yüksek ortalama fiyatlı markayı listeleyin.

```
SELECT B.BRAND_ID, B.BRAND_NAME, B.BRAND_TYPE, ROUND(AVG(P.PROD
_PRICE), 2) AS AVG_PRICE
FROM BRAND B
JOIN PRODUCT P ON B.BRAND_ID = P.BRAND_ID
GROUP BY B.BRAND_ID, B.BRAND_NAME, B.BRAND_TYPE
ORDER BY AVG_PRICE DESC
LIMIT 1;
```

47. Soyadı "Hagan" olan müşteriye 18 Mayıs 2021'de satış yapan çalışanı ve yöneticisini listeleyin.

```
SELECT MGR.EMP_FNAME AS MANAGER_FNAME, MGR.EMP_LNAME AS MA
NAGER_LNAME,
    D.DEPT_NAME, D.DEPT_PHONE, SLS.EMP_FNAME AS SALES_FNAME,
    SLS.EMP_LNAME AS SALES_LNAME, C.CUST_FNAME, C.CUST_LNAME,
    I.INV_DATE, I.INV_TOTAL
FROM CUSTOMER C
JOIN INVOICE I ON C.CUST_CODE = I.CUST_CODE
JOIN EMPLOYEE SLS ON I.EMPLOYEE_ID = SLS.EMP_NUM
JOIN DEPARTMENT D ON SLS.DEPT_NUM = D.DEPT_NUM
JOIN EMPLOYEE MGR ON D.MANAGER_ID = MGR.EMP_NUM
WHERE C.CUST_LNAME = 'Hagan' AND I.INV_DATE = '2021-05-18';
```

48. Departman 300'deki çalışanların geçerli maaşlarını listeleyin.

```
SELECT E.EMP_NUM, E.EMP_FNAME, E.EMP_LNAME, S.SAL_AMOUNT
FROM EMPLOYEE E
JOIN SALARY_HISTORY S ON E.EMP_NUM = S.EMP_NUM
WHERE E.DEPT_NUM = 300 AND S.SAL_END IS NULL
ORDER BY S.SAL_AMOUNT DESC;
```

49. Her çalışan için başlangıç maaşını listeleyin.

```
SELECT EMP_NUM, EMP_FNAME, EMP_LNAME, MIN(SAL_AMOUNT) AS STA
RTING_SALARY
FROM SALARY_HISTORY
GROUP BY EMP_NUM, EMP_FNAME, EMP_LNAME
ORDER BY EMP_NUM;
```

50. Aynı faturada aynı markanın sızdırmazlık ve son kat ürünleri

```
SELECT L1.Inv_Num, L1.Line_Num AS Sealer_Line, L2.Line_Num AS Topcoat_Li
ne,
    L1.Prod_SKU AS Sealer_SKU, L2.Prod_SKU AS Topcoat_SKU,
```

```
    L1.Prod_Descript AS Sealer_Desc, L2.Prod_Descript AS Topcoat_Desc,
    L1.Brand_ID
FROM LGLINE L1
JOIN LGLINE L2
ON L1.Inv_Num = L2.Inv_Num AND L1.Brand_ID = L2.Brand_ID
WHERE L1.Prod_Descript LIKE '%Sealer%'
AND L2.Prod_Descript LIKE '%Topcoat%'
ORDER BY L1.Inv_Num ASC, L1.Line_Num ASC, L2.Line_Num DESC;
```

51. Binder Prime markalı ürünleri en çok satan çalışan

```
SELECT E.Emp_Num, E.Emp_Fname, E.Emp_Lname, E.Emp_Email,
    SUM(L.Line_Qty) AS Total_Units_Sold
FROM LGEMPLOYEE E
JOIN LGINVOICE I ON E.Emp_Num = I.Employee_ID
JOIN LGLINE L ON I.Inv_Num = L.Inv_Num
JOIN LGPRODUCT P ON L.Prod_SKU = P.Prod_SKU
WHERE P.Brand_ID = (SELECT Brand_ID FROM LGBRAND WHERE Brand_Nam
e = 'Binder Prime')
AND I.Inv_Date BETWEEN '2021-11-01' AND '2021-12-05'
GROUP BY E.Emp_Num, E.Emp_Fname, E.Emp_Lname, E.Emp_Email
ORDER BY Total_Units_Sold DESC, E.Emp_Lname ASC;
```

52. 83649 ve 83677 çalışanları tarafından tamamlanan faturalarla müşteriler

```
SELECT DISTINCT C.Cust_Code, C.Cust_Fname, C.Cust_Lname
FROM LGCUSTOMER C
JOIN LGINVOICE I1 ON C.Cust_Code = I1.Cust_Code AND I1.Employee_ID = 83
649
JOIN LGINVOICE I2 ON C.Cust_Code = I2.Cust_Code AND I2.Employee_ID = 8
3677
ORDER BY C.Cust_Lname ASC, C.Cust_Fname ASC;
```

53. Alabama'daki müşterilerin en büyük satın alımları

```
SELECT C.Cust_Code, C.Cust_Fname, C.Cust_Lname,
      C.Cust_Street, C.Cust_City, C.Cust_State,
      I.Inv_Date, COALESCE(I.Inv_Total, 0) AS Inv_Total
FROM LGCUSTOMER C
LEFT JOIN LGINVOICE I ON C.Cust_Code = I.Cust_Code
AND C.Cust_State = 'AL'
ORDER BY C.Cust_Lname ASC, C.Cust_Fname ASC;
```

54. Markalara göre ürünlerin ortalama fiyatı ve toplam satılan birimler

```
SELECT B.Brand_Name, B.Brand_Type,
      ROUND(AVG(P.Prod_Price), 2) AS Avg_Price,
      SUM(L.Line_Qty) AS Total_Units_Sold
FROM LGBRAND B
JOIN LGPRODUCT P ON B.Brand_ID = P.Brand_ID
JOIN LGLINE L ON P.Prod_SKU = L.Prod_SKU
GROUP BY B.Brand_Name, B.Brand_Type
ORDER BY B.Brand_Name ASC;
```

55. Premium olmayan ama premium ürünlerden daha pahalı ürünler

```
SELECT B.Brand_Name, B.Brand_Type, P.Prod_SKU,
      P.Prod_Descript, P.Prod_Price
FROM LGPRODUCT P
JOIN LGBRAND B ON P.Brand_ID = B.Brand_ID
WHERE B.Brand_Type != 'Premium'
AND P.Prod_Price > (SELECT MAX(Prod_Price)
                    FROM LGPRODUCT
                    JOIN LGBRAND ON LGPRODUCT.Brand_ID = LGBRAND.Brand_ID
```

```
                WHERE Brand_Type = 'Premium')
ORDER BY P.Prod_Price DESC;
```

56. Her kitap için kitap adı, maliyet ve yayın yılı

```
SELECT Book_Title, Book_Cost, Publish_Year
FROM BOOK
ORDER BY Book_Title ASC;
```

57. Müşterilerin ad ve soyadları (büyük/küçük harfe duyarsız sıralama)

```
SELECT Cust_Fname, Cust_Lname
FROM LGCUSTOMER
ORDER BY UPPER(Cust_Lname), UPPER(Cust_Fname);
```

58. Ödünç verme bilgileri (ödünç verme numarasına göre sıralı)

```
SELECT Loan_Num, Loan_Date, Due_Date
FROM LOAN
ORDER BY Loan_Num ASC;
```

59. Kitap numarası, kitap adı ve konusu

```
SELECT Book_Num, Book_Title, Book_Subject
FROM BOOK
ORDER BY Book_Num ASC;
```

60. Farklı yayın yıllarını listeleme

```
SELECT DISTINCT Publish_Year
FROM BOOK
```

```
ORDER BY Publish_Year ASC;
```

61. Farklı kitap konuları

```
SELECT DISTINCT Book_Subject
FROM BOOK
ORDER BY Book_Subject ASC;
```

62. Kitap numarası, adı ve maliyeti

```
SELECT Book_Num, Book_Title, Book_Cost
FROM BOOK
ORDER BY Book_Num ASC;
```

63. Tüm çıkışlar: Çıkış numarası, kitap numarası, kullanıcı kimliği, çıkış ve son tarih

```
SELECT Loan_Num, Book_Num, Patron_ID, Loan_Date, Due_Date
FROM LOAN
ORDER BY Due_Date DESC, Loan_Num ASC;
```

64. Kitap adı, yılı ve konusu (birkaç kriterle sıralama)

```
SELECT Book_Title, Publish_Year, Book_Subject
FROM BOOK
ORDER BY Book_Subject ASC, Publish_Year DESC, Book_Title ASC;
```

65. Fiyatı $59.95 olan kitaplar

```
SELECT Book_Num, Book_Title, Book_Cost
FROM BOOK
```

```
WHERE Book_Cost = 59.95
ORDER BY Book_Num ASC;
```

66. "Veritabanı" konulu kitaplar

```
SELECT Book_Num, Book_Title, Book_Cost
FROM BOOK
WHERE Book_Subject = 'Veritabanı'
ORDER BY Book_Num ASC;
```

67. 5 Nisan 2021'den önce ödünç verilen kitaplar

```
SELECT Loan_Num, Book_Num, Loan_Date
FROM LOAN
WHERE Loan_Date < '2021-04-05'
ORDER BY Loan_Num ASC;
```

68. 2015 sonrası yayınlanan "Programlama" konulu kitaplar

```
SELECT Book_Num, Book_Title, Publish_Year
FROM BOOK
WHERE Publish_Year > 2015 AND Book_Subject = 'Programlama'
ORDER BY Book_Num ASC;
```

69. "Middleware" veya "Cloud" konulu ve fiyatı $70'dan fazla olan kitaplar

```
SELECT Book_Num, Book_Title, Book_Subject, Book_Cost
FROM BOOK
WHERE (Book_Subject = 'Middleware' OR Book_Subject = 'Cloud')
```

```
AND Book_Cost > 70
ORDER BY Book_Num ASC;
```

70. 1980'lerde doğan yazarlar

```
SELECT Author_ID, Author_Fname, Author_Lname, Birth_Year
FROM AUTHOR
WHERE Birth_Year BETWEEN 1980 AND 1989
ORDER BY Author_ID ASC;
```

71. Başlığında "Veritabanı" geçen kitaplar (büyük/küçük harf duyarsız)

```
SELECT Book_Num, Book_Title, Book_Subject
FROM BOOK
WHERE UPPER(Book_Title) LIKE '%VERİTABANI%'
ORDER BY Book_Num ASC;
```

72. Öğrenci olan tüm kullanıcılar

```
SELECT Patron_ID, Patron_Fname, Patron_Lname
FROM PATRON
WHERE Patron_Type = 'Student'
ORDER BY Patron_ID ASC;
```

73. Soyadı "C" harfiyle başlayan kullanıcılar

```
SELECT Patron_ID, Patron_Fname, Patron_Lname, Patron_Type
FROM PATRON
WHERE Patron_Lname LIKE 'C%'
ORDER BY Patron_ID ASC;
```

74. Doğum yılı bilinmeyen yazarlar

```
SELECT Author_ID, Author_Fname, Author_Lname
FROM AUTHOR
WHERE Birth_Year IS NULL
ORDER BY Author_ID ASC;
```

75. Doğum yılı bilinen yazarlar

```
SELECT Author_ID, Author_Fname, Author_Lname
FROM AUTHOR
WHERE Birth_Year IS NOT NULL
ORDER BY Author_ID ASC;
```

---

76. Henüz iade edilmemiş tüm ödemeler

```
SELECT Loan_Num, Book_Num, Patron_ID, Loan_Date, Due_Date
FROM LOAN
WHERE Return_Date IS NULL
ORDER BY Book_Num ASC;
```

---

77. Yazarlar (doğum yılına göre azalan, soyada göre artan sıralama)

```
SELECT Author_ID, Author_Fname, Author_Lname, Birth_Year
FROM AUTHOR
ORDER BY Birth_Year DESC, Author_Lname ASC;
```

---

78. Sistemdeki kitap sayısı

```
SELECT COUNT(*) AS Total_Books
FROM BOOK;
```

79. Farklı kitap konularının sayısı

```
SELECT COUNT(DISTINCT Book_Subject) AS Unique_Subjects
FROM BOOK;
```

80. Ödünç verilmemiş kitapların sayısı

```
SELECT COUNT(*) AS Available_Books
FROM BOOK
WHERE Book_Num NOT IN (SELECT DISTINCT Book_Num FROM LOAN);
```

81. En pahalı kitap fiyatı

```
SELECT MAX(Book_Cost) AS Max_Book_Cost
FROM BOOK;
```

82. En ucuz kitap fiyatı

```
SELECT MIN(Book_Cost) AS Min_Book_Cost
FROM BOOK;
```

83. Bir kitabı ödünç alan farklı kullanıcı sayısı

```
SELECT COUNT(DISTINCT Patron_ID) AS Unique_Borrowers
FROM LOAN;
```

84. Konu başına kitap sayısı

```
SELECT Book_Subject, COUNT(*) AS Book_Count
FROM BOOK
```

```
GROUP BY Book_Subject
ORDER BY Book_Count DESC, Book_Subject ASC;
```

85. Yazar başına yazılan kitap sayısı

```
SELECT A.Author_ID, COUNT(B.Book_Num) AS Book_Count
FROM AUTHOR A
JOIN BOOK_AUTHOR BA ON A.Author_ID = BA.Author_ID
JOIN BOOK B ON BA.Book_Num = B.Book_Num
GROUP BY A.Author_ID
ORDER BY Book_Count DESC, A.Author_ID ASC;
```

86. Kütüphanedeki tüm kitapların toplam değeri

```
SELECT SUM(Book_Cost) AS Total_Book_Value
FROM BOOK;
```

87. Her ödeme için kullanıcı, kitap ve kaç gün ödünç tutulduğu

```
SELECT Patron_ID, Book_Num, DATEDIFF(Return_Date, Loan_Date) AS Days_
Held
FROM LOAN
WHERE Return_Date IS NOT NULL
ORDER BY Days_Held DESC, Patron_ID ASC, Book_Num ASC;
```

88. Kullanıcılar ve kullanıcı türleri (kullanıcı kimliğine göre sıralı)

```
SELECT Patron_ID, CONCAT(Patron_Fname, ' ', Patron_Lname) AS Full_Name,
Patron_Type
FROM PATRON
ORDER BY Patron_ID ASC;
```

89. Kitaplar, yılları ve konuları

```
SELECT Book_Num, Publish_Year, Book_Title, Book_Subject
FROM BOOK
ORDER BY Book_Num ASC;
```

90. Yazarlar ve yazdıkları kitaplar

```
SELECT A.Author_Lname, A.Author_Fname, B.Book_Num
FROM AUTHOR A
JOIN BOOK_AUTHOR BA ON A.Author_ID = BA.Author_ID
JOIN BOOK B ON BA.Book_Num = B.Book_Num
ORDER BY A.Author_Lname ASC, A.Author_Fname ASC, B.Book_Num ASC;
```

91. Her kitap için yazarlar ve kitap bilgileri

```
SELECT A.Author_ID, B.Book_Num, B.Book_Title, B.Book_Subject
FROM AUTHOR A
JOIN BOOK_AUTHOR BA ON A.Author_ID = BA.Author_ID
JOIN BOOK B ON BA.Book_Num = B.Book_Num
ORDER BY B.Book_Num ASC, A.Author_ID ASC;
```

92. Yazarlar, kitap adları ve değiştirme maliyetleri

```
SELECT A.Author_Lname, A.Author_Fname, B.Book_Title, B.Book_Cost
FROM AUTHOR A
JOIN BOOK_AUTHOR BA ON A.Author_ID = BA.Author_ID
JOIN BOOK B ON BA.Book_Num = B.Book_Num
ORDER BY B.Book_Num ASC, A.Author_ID ASC;
```

93. Şu anda ödünç verilmiş kitaplar

```
SELECT L.Patron_ID, B.Book_Num, CONCAT(P.Patron_Fname, ' ', P.Patron_Lna
me) AS Patron_Name, B.Book_Title
FROM LOAN L
JOIN BOOK B ON L.Book_Num = B.Book_Num
JOIN PATRON P ON L.Patron_ID = P.Patron_ID
WHERE L.Return_Date IS NULL
ORDER BY P.Patron_Lname ASC, B.Book_Title ASC;
```

---

94. Kullanıcılar (tam ad ve türüne göre sıralama)

```
SELECT Patron_ID, CONCAT(Patron_Fname, ' ', Patron_Lname) AS Full_Name,
Patron_Type
FROM PATRON
ORDER BY Patron_Type ASC, Patron_Fname ASC;
```

---

95. Kaç kez ödünç verilen kitaplar (hiç verilmemiş olanlar hariç)

```
SELECT Book_Num, COUNT(*) AS Loan_Count
FROM LOAN
GROUP BY Book_Num
HAVING COUNT(*) > 0
ORDER BY Loan_Count DESC, Book_Num DESC;
```

---

96. "Bulut" konulu kitaplar ve yazarları

```
SELECT A.Author_ID, A.Author_Fname, A.Author_Lname, B.Book_Num, B.Book
_Title
FROM BOOK B
JOIN BOOK_AUTHOR BA ON B.Book_Num = BA.Book_Num
JOIN AUTHOR A ON BA.Author_ID = A.Author_ID
WHERE B.Book_Subject = 'Bulut'
ORDER BY B.Book_Title ASC, A.Author_Lname ASC;
```

97. En çok kitap ödünç alan kullanıcılar

```
SELECT Patron_ID, COUNT(Book_Num) AS Loan_Count
FROM LOAN
GROUP BY Patron_ID
ORDER BY Loan_Count DESC, Patron_ID ASC;
```

98. 2021'de ödünç alınan kitaplar

```
SELECT Book_Num, Loan_Date
FROM LOAN
WHERE YEAR(Loan_Date) = 2021
ORDER BY Loan_Date ASC;
```

99. Aynı anda 2'den fazla kitap ödünç alan kullanıcılar

```
SELECT Patron_ID, COUNT(Book_Num) AS Loan_Count
FROM LOAN
GROUP BY Patron_ID, Loan_Date
HAVING COUNT(Book_Num) > 2
ORDER BY Loan_Count DESC;
```

100. Kullanıcı başına toplam borç

```
SELECT Patron_ID, SUM(Book_Cost) AS Total_Cost
FROM LOAN L
JOIN BOOK B ON L.Book_Num = B.Book_Num
GROUP BY Patron_ID
ORDER BY Total_Cost DESC;
```

101. 2021'de ödünç alınan ve iade edilen kitaplar

```
SELECT Book_Num, Loan_Date, Return_Date
FROM LOAN
WHERE YEAR(Loan_Date) = 2021 AND Return_Date IS NOT NULL
ORDER BY Loan_Date ASC;
```

102. Bir kitap için tüm ödünç alma kayıtları

```
SELECT Book_Num, Patron_ID, Loan_Date, Return_Date
FROM LOAN
WHERE Book_Num = 'B001'
ORDER BY Loan_Date ASC;
```

103. 2020’den beri hiç ödünç verilmemiş kitaplar

```
SELECT Book_Num, Book_Title
FROM BOOK
WHERE Book_Num NOT IN (
   SELECT DISTINCT Book_Num FROM LOAN WHERE Loan_Date >= '2020-01
-01'
)
ORDER BY Book_Num ASC;
```

104. Aynı kullanıcı tarafından birden fazla kez ödünç alınan kitaplar

```
SELECT Book_Num, Patron_ID, COUNT(*) AS Loan_Count
FROM LOAN
GROUP BY Book_Num, Patron_ID
HAVING COUNT(*) > 1
ORDER BY Loan_Count DESC;
```

105. Her konu için ortalama kitap maliyeti

```
SELECT Book_Subject, ROUND(AVG(Book_Cost), 2) AS Avg_Cost
FROM BOOK
GROUP BY Book_Subject
ORDER BY Avg_Cost DESC;
```

106. En son ödünç alınan kitaplar

```
SELECT Book_Num, Loan_Date
FROM LOAN
ORDER BY Loan_Date DESC
LIMIT 5;
```

107. Bir kullanıcının ödünç aldığı en pahalı kitap

```
SELECT Patron_ID, MAX(Book_Cost) AS Max_Cost
FROM LOAN L
JOIN BOOK B ON L.Book_Num = B.Book_Num
GROUP BY Patron_ID
ORDER BY Max_Cost DESC;
```

108. Bir kullanıcının ödünç aldığı kitap sayısı

```
SELECT Patron_ID, COUNT(Book_Num) AS Book_Count
FROM LOAN
GROUP BY Patron_ID
ORDER BY Book_Count DESC;
```

109. En çok ödünç alınan konu başlığı

```
SELECT B.Book_Subject, COUNT(L.Book_Num) AS Loan_Count
FROM LOAN L
JOIN BOOK B ON L.Book_Num = B.Book_Num
GROUP BY B.Book_Subject
ORDER BY Loan_Count DESC
LIMIT 1;
```
