📊 Kredi Riski Verileriyle Uygulamalı SQL - Tuğçe Şahin

Temel Seviye Konu Anlatımı

1.1. SELECT * ile Tüm Sütunları Çekme

Konu: `SELECT *` komutu, bir tablodaki tüm sütunları ve satırları getirir. Tablonun yapısını tanımak veya tüm veriyi görmek istediğimizde kullanılır.

Örnek:

```
SELECT * FROM customers; -- Bu sorgu customers tablosundaki tüm satır ve sütunları getirir.
```

Soru: defaults tablosundaki tüm sütunları ve satırları getiren sorguyu yazın.

Cevap: SELECT * FROM defaults;

1.2. Belirli Sütunları Çekmek

Konu: `SELECT sütun1, sütun2` şeklinde kullanarak sadece ihtiyacınız olan bilgileri alabilirsiniz. Bu, gereksiz verileri elemine eder.

Örnek:

```
SELECT name, city FROM customers; -- Sadece müşteri adları ve şehirleri gelir.
```

Soru: loans tablosundan yalnızca kredi numarası (loan_id) ve durum (status) sütunlarını çekin.

Cevap: SELECT loan_id, status FROM loans;

1.3. SELECT AS ile Takma Ad (Alias) Kullanmak

Konu: `AS` komutu sütunlara geçici takma adlar verir. Raporlar veya kullanıcı dostu çıktılar hazırlamak için kullanılır.

Örnek:

```
SELECT income AS gelir, city AS sehir FROM customers;
```

Soru: payments tablosundaki `amount_paid` sütununu 'odenen_tutar' adıyla listeleyin.

Cevap: SELECT amount_paid AS odenen_tutar FROM payments;

1.4. SELECT ile Matematiksel İşlemler

Konu: Sayılarla çalışan sütunlar üzerinde toplama (+), çıkarma (-), çarpma (*), bölme (/) gibi işlemler yapabilirsiniz.

Örnek:

```
SELECT income, income * 12 AS yillik_gelir FROM customers;
```

Soru: loans tablosundaki `amount` değerlerinin yarısını (amount / 2) hesaplayıp 'yarisi' takma adıyla listeleyin.

Cevap: SELECT amount / 2 AS yarisi FROM loans;

1.5. SELECT DISTINCT ile Tekil Değerleri Almak

Konu: `DISTINCT` ifadesi aynı değere sahip satırları eleyerek her benzersiz (eşsiz) değeri yalnızca bir kez getirir.

Örnek:

```
SELECT DISTINCT city FROM customers;
```

Soru: customers tablosundaki tekil medeni durumları (`marital_status`) listeleyen sorguyu yazın.

Cevap: SELECT DISTINCT marital_status FROM customers;

2.1. FROM ile Tabloyu Belirtmek

Konu: `FROM` ifadesi, verinin hangi tablodan çekileceğini belirtir. `SELECT` komutunun hangi kaynakla çalışacağını belirlemek için kullanılır.

Örnek:

```
SELECT income FROM customers; -- customers tablosundan income sütunu alınır.
```

Soru: defaults tablosundaki tüm verileri getiren SQL sorgusunu yazın.

Cevap: SELECT * FROM defaults;

2.2. FROM ile Tabloya Takma Ad (Alias) Verme

Konu: `AS` ifadesi, bir tabloya kısa bir takma ad vererek yazım kolaylığı sağlar. Genellikle JOIN işlemlerinde veya çoklu tablo içeren sorgularda tercih edilir.

Örnek:

```
SELECT c.name FROM customers AS c; -- customers tablosu 'c' olarak kısaltıldı.
```

Soru: payments tablosuna 'p' takma adı vererek `amount_paid` değerlerini listeleyen SQL sorgusunu yazın.

Cevap: SELECT p.amount_paid FROM payments AS p;

3.1. WHERE ile Basit Filtreleme

Konu: `WHERE` komutu, sadece belirli bir koşulu sağlayan satırları seçmek için kullanılır. Örneğin, bir müşterinin geliri belli bir değerin üzerindeyse sadece onu listeleyebilirsiniz.

Örnek:

```
SELECT name FROM customers WHERE income > 6000; -- Geliri 6000’den fazla olan müşteriler gelir.
```

Soru: loans tablosunda durumu (status) 'unpaid' olan kredileri listeleyin.

Cevap: SELECT * FROM loans WHERE status = 'unpaid';

3.2. WHERE ile Çoklu Koşullar: AND & OR

Konu: `AND` ifadesiyle birden fazla koşul aynı anda kontrol edilir. `OR` ile koşullardan herhangi biri sağlanıyorsa satır seçilir.

Örnek:

```
SELECT name FROM customers WHERE income > 4000 AND city = 'Istanbul'; -- Geliri 4000'den fazla ve İstanbul'da yaşayanlar gelir.
```

Soru: credit_score'u 700’den büyük **veya** doğum yılı 1995 olan müşterileri listeleyin.

Cevap: SELECT * FROM customers WHERE credit_score > 700 OR birth_year = 1995;

3.3. Gelişmiş Filtreleme: BETWEEN, IN, NOT IN

Konu: `BETWEEN` ile iki değer arasında kalan veriler alınır. `IN` ile birden fazla sabit değeri kapsayan filtreleme yapılır. `NOT IN` ile belirtilen değerler hariç tutulur.

Örnek:

```
SELECT name FROM customers WHERE birth_year BETWEEN 1985 AND 1990; -- 1985-1990 arası doğanları getirir.
```

Soru: loans tablosunda miktarı (amount) 8000 veya 15000 olan kredileri listeleyin.

Cevap: SELECT * FROM loans WHERE amount IN (8000, 15000);

3.4. Boş Değerleri Kontrol Etmek: IS NULL / IS NOT NULL

Konu: SQL'de `NULL` boş veri demektir. Bu verileri filtrelemek için `IS NULL` veya `IS NOT NULL` ifadeleri kullanılır.

Örnek:

```
SELECT * FROM customers WHERE job_title IS NOT NULL; -- job_title bilgisi olanları getirir.
```

Soru: defaults tablosunda reason sütunu boş olmayan kayıtları listeleyin.

Cevap: SELECT * FROM defaults WHERE reason IS NOT NULL;

4.1. Artan Sıralama: ORDER BY ASC

Konu: `ORDER BY sütun ASC` ifadesi, verileri o sütuna göre küçükten büyüğe sıralar. `ASC` yazılmasa bile varsayılan olarak zaten artan sıralama uygulanır.

Örnek:

```
SELECT name FROM customers ORDER BY credit_score ASC; -- Kredi skoruna göre küçükten büyüğe sıralar.
```

Soru: payments tablosundaki ödenen tutarları artan şekilde sıralayarak listeleyin.

Cevap: SELECT * FROM payments ORDER BY amount_paid ASC;

4.2. Azalan Sıralama: ORDER BY DESC

Konu: `ORDER BY sütun DESC` ifadesi, verileri büyükten küçüğe sıralar. Özellikle en yüksek geliri, en büyük kredi miktarını vb. listelemek için kullanılır.

Örnek:

```
SELECT name FROM customers ORDER BY income DESC; -- En yüksek gelir en üstte gelir.
```

Soru: loans tablosundaki kredi tutarlarını azalan şekilde sıralayın.

Cevap: SELECT * FROM loans ORDER BY amount DESC;

4.3. Çoklu Sıralama: ORDER BY sütun1, sütun2

Konu: `ORDER BY sütun1, sütun2` şeklinde sıralama yapılırsa önce birinci sütuna göre sıralanır, eğer eşitlik varsa ikinci sütun dikkate alınır.

Örnek:

```
SELECT * FROM customers ORDER BY marital_status, income DESC; -- Önce medeni duruma göre, sonra gelire göre sırala.
```

Soru: customers tablosunu şehir adına göre artan, ardından doğum yılına göre azalan şekilde sıralayın.

Cevap: SELECT * FROM customers ORDER BY city ASC, birth_year DESC;

5.1. LIMIT ile İlk n Kayıdı Getirme

Konu: `LIMIT` komutu ile sadece belirli sayıda kayıt getirilebilir. Bu, genellikle verinin ilk kısmına bakmak istediğimizde kullanılır.

Örnek:

```
SELECT * FROM customers LIMIT 2; -- İlk 2 müşteri getirilir.
```

Soru: defaults tablosundaki yalnızca ilk kaydı çekin.

Cevap: SELECT * FROM defaults LIMIT 1;

5.2. LIMIT ve ORDER BY ile En Yüksek Değerli İlk Kayıt

Konu: `ORDER BY` ile veriler sıralanır, ardından `LIMIT` ile sadece ilk sonuçlar getirilir. Bu, örneğin en yüksek miktarlı krediyi çekmek için idealdir.

Örnek:

```
SELECT * FROM customers ORDER BY income DESC LIMIT 1; -- En yüksek gelire sahip müşteri getirilir.
```

Soru: loans tablosundaki en küçük tutarlı krediyi çekin.

Cevap: SELECT * FROM loans ORDER BY amount ASC LIMIT 1;

6.1. INNER JOIN ile Tablo Birleştirme

Konu: `INNER JOIN` ile iki tablo ortak bir sütuna göre birleştirilir ve sadece eşleşen satırlar alınır.

Örnek:

```
SELECT name, amount FROM customers JOIN loans ON customers.customer_id = loans.customer_id;
```

Soru: payment yapılan krediler için müşteri adını ve ödeme tutarını gösterin.

Cevap: SELECT name, amount_paid FROM customers JOIN loans ON customers.customer_id = loans.customer_id JOIN payments ON loans.loan_id = payments.loan_id;

6.2. LEFT JOIN ile Eşleşmeyen Verileri Dahil Etme

Konu: `LEFT JOIN` ile birleştirme yapılırken sol tablodaki veriler her koşulda alınır. Sağdaki eşleşmeyen verilerde boş (`NULL`) görünebilir.

Örnek:

```
SELECT name, reason FROM customers LEFT JOIN defaults ON customers.customer_id = defaults.customer_id;
```

Soru: defaults tablosundaki verileri, müşterilerle birleştirerek gösterin. Eşleşmeyen müşteri olursa da gösterilsin.

Cevap: SELECT d.customer_id, name FROM defaults d LEFT JOIN customers c ON d.customer_id = c.customer_id;

6.3. JOIN Kullanırken Alias (Takma Ad) Kullanımı

Konu: `AS` ile tablolara kısa takma adlar (alias) vererek okunabilirliği artırabiliriz. `c.name`, `l.amount` gibi yazmak daha kısa ve temiz olur.

Örnek:

```
SELECT c.name, l.amount FROM customers AS c JOIN loans AS l ON c.customer_id = l.customer_id;
```

Soru: alias kullanarak müşteri adları ve kredi durumunu listeleyin.

Cevap: SELECT c.name, l.status FROM customers AS c JOIN loans AS l ON c.customer_id = l.customer_id;