PL/SQL Kapsamlı Pratik Bankası

Her Konu İçin 9 Soru - Toplam 36 Soru

3 Kolay • 3 Orta • 2 Zor • 1 Sınav Sorusu

KONU 1: Temel PL/SQL Yapıları

Değişkenler, Blok Yapıları, Veri Tipleri, Operatörler

Soru 1 [KOLAY] - Basit String Değişkeni

Bir çalışanın adını ve soyadını ayrı değişkenlerde tutup, tam adı olarak birleştirerek ekrana yazdırın.

ÇÖZÜM:

```
DECLARE
    v_first_name VARCHAR2(50) := 'Mehmet';
    v_last_name VARCHAR2(50) := 'Yılmaz';
    v_full_name VARCHAR2(100);
BEGIN
    v_full_name := v_first_name || ' ' || v_last_name;
    DBMS_OUTPUT.PUT_LINE('Tam Ad: ' || v_full_name);
END;
```

AÇIKLAMA:

VARCHAR2 string veri tipi için kullanılır. || operatörü stringleri birleştirir. := atama operatörüdür. v_full_name değişkeni hesaplanan değeri tutar. DBMS_OUTPUT.PUT_LINE ekrana yazdırır (SET SERVEROUTPUT ON gerekli).

Soru 2 [KOLAY] - Sayısal İşlemler

İki sayıyı değişkenlere atayın, toplam, fark, çarpım ve bölümlerini hesaplayıp yazdırın.

ÇÖZÜM:

```
DECLARE
    v_sayi1 NUMBER := 100;
    v_sayi2 NUMBER := 25;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Toplam: ' || (v_sayi1 + v_sayi2));
    DBMS_OUTPUT.PUT_LINE('Fark: ' || (v_sayi1 - v_sayi2));
    DBMS_OUTPUT.PUT_LINE('Çarpım: ' || (v_sayi1 * v_sayi2));
    DBMS_OUTPUT.PUT_LINE('Bölüm: ' || (v_sayi1 / v_sayi2));
END;
```

AÇIKLAMA:

NUMBER veri tipi tüm sayısal değerler için kullanılır. Temel aritmetik operatörler: + (toplama), - (çıkarma), * (çarpma), / (bölme). Parantez içinde işlemler yapılır ve sonuç string ile birleştirilir.

Soru 3 [KOLAY] - Sabit (CONSTANT) Kullanımı

Bir daire için yarıçapı 5 cm olarak sabit tanımlayın, alan ve çevreyi hesaplayın (π=3.14).

ÇÖZÜM:

```
DECLARE
    c_pi CONSTANT NUMBER := 3.14;
    c_radius CONSTANT NUMBER := 5;
    v_area NUMBER;
    v_circumference NUMBER;
BEGIN
    v_area := c_pi * c_radius * c_radius;
    v_circumference := 2 * c_pi * c_radius;

    DBMS_OUTPUT.PUT_LINE('Alan: ' || v_area || ' cm²');
    DBMS_OUTPUT.PUT_LINE('Çevre: ' || v_circumference || ' cm');
END;
```

AÇIKLAMA:

CONSTANT anahtar kelimesi değişmeyen değerler için kullanılır. c_ öneki constant olduğunu gösterir (naming convention). Constant değişkenler değiştirilemez. Alan formülü: π*r², Çevre formülü: 2*π*r.

Soru 4 [ORTA] - İç İçe Bloklar

Dış blokta bir sayı tanımlayın (x=10). İç blokta aynı isimde başka bir sayı tanımlayın (x=20). Her ikisini de yazdırın.

ÇÖZÜM:

```
<<outer_block>>
DECLARE
    x NUMBER := 10;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Dış blok x: ' || x);

    <<inner_block>>
    DECLARE
        x NUMBER := 20;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('İç blok x: ' || x);
        DBMS_OUTPUT.PUT_LINE('Dış blok x (etiketle): ' ||
outer_block.x);
    END inner_block;

    DBMS_OUTPUT.PUT_LINE('Dış blok x (tekrar): ' || x);
END outer_block;
```

AÇIKLAMA:

Blok etiketleri <<etiket_adi>> şeklinde tanımlanır. İç bloktaki x değişkeni dış bloktaki x değişkenini gölgeler (shadowing). outer_block.x ile dış bloktaki değişkene erişilir. Scope kuralları: iç blok dış bloğu görür ama tersi geçerli değildir.

Soru 5 [ORTA] - DATE Veri Tipi

Bugünün tarihini alın, 30 gün sonrasını hesaplayın ve ikisini de yazdırın.

ÇÖZÜM:

```
DECLARE
    v_today DATE;
    v_future DATE;
    v_diff NUMBER;
BEGIN
    v_today := SYSDATE;
    v_future := v_today + 30;
```

```
    v_diff := v_future - v_today;

    DBMS_OUTPUT.PUT_LINE('Bugün: ' || TO_CHAR(v_today, 'DD-MON-
YYYY'));
    DBMS_OUTPUT.PUT_LINE('30 gün sonra: ' || TO_CHAR(v_future, 'DD-
MON-YYYY'));
    DBMS_OUTPUT.PUT_LINE('Fark: ' || v_diff || ' gün');
END;
```

AÇIKLAMA:

SYSDATE sistem tarihini döndürür. DATE tipine +/- yaparak gün ekleyip çıkarabiliriz. İki DATE farkı gün sayısı verir. TO_CHAR tarihi formatlı stringe çevirir. DD=gün, MON=ay (kısa), YYYY=yıl formatları.

Soru 6 [ORTA] - BOOLEAN Mantığı

Bir kişinin yaşını alın (25), 18 yaş üstü mü ve 65 yaş altı mı kontrol edin, çalışabilir mi değerlendirin.

ÇÖZÜM:

```
DECLARE
    v_age NUMBER := 25;
    v_is_adult BOOLEAN;
    v_not_retired BOOLEAN;
    v_can_work BOOLEAN;
BEGIN
    v_is_adult := (v_age >= 18);
    v_not_retired := (v_age < 65);
    v_can_work := v_is_adult AND v_not_retired;

    DBMS_OUTPUT.PUT_LINE('Yaş: ' || v_age);
    IF v_can_work THEN
        DBMS_OUTPUT.PUT_LINE('Durum: Çalışabilir');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Durum: Çalışamaz');
    END IF;
END;
```

AÇIKLAMA:

BOOLEAN TRUE, FALSE veya NULL değer alır. Karşılaştırma operatörleri (>=, <) BOOLEAN döndürür. AND operatörü her iki koşul da TRUE ise TRUE döner. BOOLEAN değerler direkt yazdırılamaz, IF ile kontrol edilmelidir.

Soru 7 [ZOR] - %TYPE ile Dinamik Tip

Employees tablosundan bir employee_id seçin ve o çalışanın tüm bilgilerini %TYPE kullanarak değişkenlere atayın.

ÇÖZÜM:

```
DECLARE
    v_emp_id employees.employee_id%TYPE := 100;
    v_first_name employees.first_name%TYPE;
    v_last_name employees.last_name%TYPE;
    v_salary employees.salary%TYPE;
    v_hire_date employees.hire_date%TYPE;
BEGIN
    SELECT first_name, last_name, salary, hire_date
    INTO v_first_name, v_last_name, v_salary, v_hire_date
    FROM employees
    WHERE employee_id = v_emp_id;
```

```
    DBMS_OUTPUT.PUT_LINE('Çalışan: ' || v_first_name || ' ' ||
v_last_name);
    DBMS_OUTPUT.PUT_LINE('Maaş: $' || v_salary);
    DBMS_OUTPUT.PUT_LINE('İşe Giriş: ' || TO_CHAR(v_hire_date, 'DD-
MON-YYYY'));
END;
```

AÇIKLAMA:

%TYPE bir kolonun veri tipini otomatik alır. employees.salary%TYPE salary kolonuyla aynı tipte değişken yaratır. Veri tipi değişirse kod otomatik uyum sağlar (maintenance kolaylığı). SELECT INTO ile birden fazla değer aynı anda atanabilir.

Soru 8 [ZOR] - Karmaşık Hesaplama

Bir çalışanın brüt maaşı 5000$. %18 gelir vergisi, %15 SGK kesintisi olsun. Net maaşı ve toplam kesinti yüzdesini hesaplayın.

ÇÖZÜM:

```
DECLARE
    c_gross_salary CONSTANT NUMBER := 5000;
    c_tax_rate CONSTANT NUMBER := 18;
    c_sgk_rate CONSTANT NUMBER := 15;
    v_tax_amount NUMBER;
    v_sgk_amount NUMBER;
    v_total_deduction NUMBER;
    v_net_salary NUMBER;
    v_total_rate NUMBER;
BEGIN
    v_tax_amount := c_gross_salary * c_tax_rate / 100;
    v_sgk_amount := c_gross_salary * c_sgk_rate / 100;
    v_total_deduction := v_tax_amount + v_sgk_amount;
    v_net_salary := c_gross_salary - v_total_deduction;
    v_total_rate := (v_total_deduction / c_gross_salary) * 100;

    DBMS_OUTPUT.PUT_LINE('Brüt Maaş: $' || c_gross_salary);
    DBMS_OUTPUT.PUT_LINE('Gelir Vergisi (%' || c_tax_rate || '): $'
|| v_tax_amount);
    DBMS_OUTPUT.PUT_LINE('SGK Kesintisi (%' || c_sgk_rate || '): $'
|| v_sgk_amount);
    DBMS_OUTPUT.PUT_LINE('Toplam Kesinti: $' || v_total_deduction ||
' (%' || ROUND(v_total_rate, 2) || ')');
    DBMS_OUTPUT.PUT_LINE('Net Maaş: $' || v_net_salary);
END;
```

AÇIKLAMA:

Karmaşık iş hesaplamalarında her adımı ayrı değişkende tutmak okunabilirlik sağlar. ROUND(sayı, 2) iki ondalık basamağa yuvarlar. Yüzde hesabı: (kısım/bütün)*100. Constant değerler değişmeyecek parametreler için idealdir.

Soru 9 [SINAV] - Temel Yapılar Kapsamlı

Kullanıcıdan iki sayı alın (&), bu sayıların toplamını, farkını ve ortalamasını hesaplayın. Ortalama 50'den büyükse "Yüksek", değilse "Düşük" yazdırın.

ÇÖZÜM:

```
DECLARE
    v_sayi1 NUMBER;
```

```
    v_sayi2 NUMBER;
    v_toplam NUMBER;
    v_fark NUMBER;
    v_ortalama NUMBER;
BEGIN
    v_sayi1 := &sayi1;
    v_sayi2 := &sayi2;

    v_toplam := v_sayi1 + v_sayi2;
    v_fark := v_sayi1 - v_sayi2;
    v_ortalama := v_toplam / 2;

    DBMS_OUTPUT.PUT_LINE('Toplam: ' || v_toplam);
    DBMS_OUTPUT.PUT_LINE('Fark: ' || v_fark);
    DBMS_OUTPUT.PUT_LINE('Ortalama: ' || v_ortalama);

    IF v_ortalama > 50 THEN
        DBMS_OUTPUT.PUT_LINE('Sonuç: Yüksek');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Sonuç: Düşük');
    END IF;
END;
```

AÇIKLAMA:

& operatörü runtime'da kullanıcıdan değer alır. Değişken tanımlama, aritmetik işlemler, IF-ELSE ve DBMS_OUTPUT kullanımı birleştirilmiş. Bu tipik bir sınav sorusudur çünkü birden fazla konsepti test eder.

KONU 2: Koşullar ve Döngüler

IF-ELSIF-ELSE, CASE, FOR LOOP, WHILE LOOP

Soru 10 [KOLAY] - Basit IF-ELSE

Bir sayının pozitif, negatif veya sıfır olduğunu kontrol edin.

ÇÖZÜM:

```
DECLARE
    v_number NUMBER := -5;
BEGIN
    IF v_number > 0 THEN
        DBMS_OUTPUT.PUT_LINE(v_number || ' pozitif bir sayıdır');
    ELSIF v_number < 0 THEN
        DBMS_OUTPUT.PUT_LINE(v_number || ' negatif bir sayıdır');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Sayı sıfırdır');
    END IF;
END;
```

AÇIKLAMA:

IF-ELSIF-ELSE yapısı birden fazla koşulu sırayla kontrol eder. İlk TRUE olan çalışır, geri kalan kontrol edilmez. DİKKAT: ELSIF yazımı (E olmadan), ELSEIF YANLIŞ! > (büyük), < (küçük) karşılaştırma operatörleridir.

Soru 11 [KOLAY] - Tek/Çift Kontrolü

Bir sayının tek mi çift mi olduğunu MOD fonksiyonu ile bulun.

ÇÖZÜM:

```
DECLARE
    v_number NUMBER := 17;
BEGIN
    IF MOD(v_number, 2) = 0 THEN
        DBMS_OUTPUT.PUT_LINE(v_number || ' çift sayıdır');
    ELSE
        DBMS_OUTPUT.PUT_LINE(v_number || ' tek sayıdır');
    END IF;
END;
```

AÇIKLAMA:

MOD(sayı, bölen) fonksiyonu modül (kalan) hesaplar. MOD(17, 2) = 1 (17/2 = 8 kalan 1). Çift sayılar 2'ye bölündüğünde kalan 0 olur. = operatörü eşitlik kontrolüdür.

Soru 12 [KOLAY] - Basit FOR LOOP

1'den 10'a kadar sayıları ekrana yazdırın.

ÇÖZÜM:

```
BEGIN
    FOR i IN 1..10 LOOP
        DBMS_OUTPUT.PUT_LINE('Sayı: ' || i);
    END LOOP;
END;
```

AÇIKLAMA:

FOR i IN başlangıç..bitiş LOOP döngü yaratır. i değişkeni otomatik tanımlanır, DECLARE gerekmez. 1..10 aralığı 1,2,3...10 demektir (dahil). Her iterasyonda LOOP-END LOOP arası çalışır.

Soru 13 [ORTA] - İç İçe IF

Bir öğrencinin notu (0-100) için: 90+ AA, 85-89 BA, 80-84 BB, 70-79 CB, 60-69 CC, 50-59 DC, 45-49 DD, 40-44 FD, <40 FF yazdırın.

ÇÖZÜM:

```
DECLARE
    v_grade NUMBER := 73;
    v_letter VARCHAR2(2);
BEGIN
    IF v_grade >= 90 THEN
        v_letter := 'AA';
    ELSIF v_grade >= 85 THEN
        v_letter := 'BA';
    ELSIF v_grade >= 80 THEN
        v_letter := 'BB';
    ELSIF v_grade >= 70 THEN
        v_letter := 'CB';
    ELSIF v_grade >= 60 THEN
        v_letter := 'CC';
    ELSIF v_grade >= 50 THEN
        v_letter := 'DC';
    ELSIF v_grade >= 45 THEN
        v_letter := 'DD';
    ELSIF v_grade >= 40 THEN
        v_letter := 'FD';
    ELSE
        v_letter := 'FF';
    END IF;

    DBMS_OUTPUT.PUT_LINE('Not: ' || v_grade || ' -> Harf: ' ||
v_letter);
END;
```

AÇIKLAMA:

Çoklu ELSIF yapısı birçok aralığı kontrol eder. Yukarıdan aşağı kontrol edilir, ilk TRUE olan çalışır. Sıralama önemli: önce 90, sonra 85, sonra 80... >= (büyük eşit) operatörü kullanılır.

Soru 14 [ORTA] - CASE İfadesi

Bir departman koduna (10, 20, 30, 40) göre departman ismini CASE ile bulun.

ÇÖZÜM:

```
DECLARE
    v_dept_id NUMBER := 20;
    v_dept_name VARCHAR2(50);
BEGIN
    v_dept_name := CASE v_dept_id
        WHEN 10 THEN 'Administration'
        WHEN 20 THEN 'Marketing'
        WHEN 30 THEN 'Purchasing'
        WHEN 40 THEN 'Human Resources'
        ELSE 'Unknown Department'
```

```
    END;

    DBMS_OUTPUT.PUT_LINE('Departman ' || v_dept_id || ': ' ||
v_dept_name);
END;
```

AÇIKLAMA:

CASE ifadesi değer döndürür (IF-ELSE statement'tır, değer döndürmez). Simple CASE: CASE değişken WHEN değer THEN sonuç formatı. ELSE opsiyoneldir ama tavsiye edilir (default değer). END ile kapatılır.

Soru 15 [ORTA] - WHILE LOOP

1'den başlayarak toplam 100'ü geçene kadar sayıları toplayın, kaç sayı toplandığını bulun.

ÇÖZÜM:

```
DECLARE
    v_sum NUMBER := 0;
    v_counter NUMBER := 1;
BEGIN
    WHILE v_sum < 100 LOOP
        v_sum := v_sum + v_counter;
        v_counter := v_counter + 1;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Toplam: ' || v_sum);
    DBMS_OUTPUT.PUT_LINE('Toplanan sayı adedi: ' || (v_counter - 1));
END;
```

AÇIKLAMA:

WHILE koşul LOOP: koşul TRUE olduğu sürece döner. Koşul başta kontrol edilir (önce kontrol, sonra çalıştır). v_counter artırılmazsa sonsuz döngüye girer! Son değer 1 fazla olur çünkü koşul sonra kontrol edilir.

Soru 16 [ZOR] - Nested LOOP ve EXIT

Çarpım tablosu oluşturun (1x1'den 10x10'a kadar). 5x5'ten sonra durdurun.

ÇÖZÜM:

```
DECLARE
    v_result NUMBER;
BEGIN
    FOR i IN 1..10 LOOP
        FOR j IN 1..10 LOOP
            v_result := i * j;
            DBMS_OUTPUT.PUT_LINE(i || ' x ' || j || ' = ' ||
v_result);

            IF i = 5 AND j = 5 THEN
                DBMS_OUTPUT.PUT_LINE('5x5'e ulaşıldı,
durduruluyor...');
                EXIT;  -- İç döngüden çık
            END IF;
        END LOOP;

        EXIT WHEN i = 5;  -- Dış döngüden çık
    END LOOP;
```

```
    DBMS_OUTPUT.PUT_LINE('Program tamamlandı');
END;
```

AÇIKLAMA:

İç içe döngüler: dış döngünün her iterasyonunda iç döngü tamamen çalışır. EXIT komutu döngüden çıkar. EXIT WHEN koşul: koşul TRUE ise çık. EXIT sadece içinde olduğu döngüden çıkar (inner loop EXIT outer loop'u etkilemez).

Soru 17 [ZOR] - Searched CASE ve Karmaşık Mantık

BMI hesaplama: Kilo ve boy alın, BMI'ı hesaplayın ve kategoriye ayırın (<18.5 Zayıf, 18.5-24.9 Normal, 25-29.9 Fazla Kilolu, 30+ Obez).

ÇÖZÜM:

```
DECLARE
    v_weight NUMBER := 75;  -- kg
    v_height NUMBER := 1.75;  -- metre
    v_bmi NUMBER;
    v_category VARCHAR2(20);
BEGIN
    v_bmi := v_weight / (v_height * v_height);

    v_category := CASE
        WHEN v_bmi < 18.5 THEN 'Zayıf'
        WHEN v_bmi < 25 THEN 'Normal'
        WHEN v_bmi < 30 THEN 'Fazla Kilolu'
        ELSE 'Obez'
    END;

    DBMS_OUTPUT.PUT_LINE('Boy: ' || v_height || ' m');
    DBMS_OUTPUT.PUT_LINE('Kilo: ' || v_weight || ' kg');
    DBMS_OUTPUT.PUT_LINE('BMI: ' || ROUND(v_bmi, 2));
    DBMS_OUTPUT.PUT_LINE('Kategori: ' || v_category);
END;
```

AÇIKLAMA:

Searched CASE: CASE WHEN koşul THEN değer formatı. Simple CASE'den farkı: koşul yazılır (değer değil). BMI formülü: kilo/(boy²). ROUND(v_bmi, 2) iki ondalık basamağa yuvarlar. Karmaşık hesaplama + koşul birlikte kullanılmış.

Soru 18 [SINAV] - Kapsamlı Döngü ve Koşul

FOR LOOP ile 1'den 20'ye kadar sayıları kontrol edin: Hem 3 hem 5'e bölünenleri "FizzBuzz", sadece 3'e bölünenleri "Fizz", sadece 5'e bölünenleri "Buzz", diğerlerini olduğu gibi yazdırın.

ÇÖZÜM:

```
BEGIN
    FOR i IN 1..20 LOOP
        IF MOD(i, 3) = 0 AND MOD(i, 5) = 0 THEN
            DBMS_OUTPUT.PUT_LINE(i || ': FizzBuzz');
        ELSIF MOD(i, 3) = 0 THEN
            DBMS_OUTPUT.PUT_LINE(i || ': Fizz');
        ELSIF MOD(i, 5) = 0 THEN
            DBMS_OUTPUT.PUT_LINE(i || ': Buzz');
        ELSE
            DBMS_OUTPUT.PUT_LINE(i || ': ' || i);
        END IF;
    END LOOP;
```

```
END;
```

AÇIKLAMA:

Klasik FizzBuzz problemi. MOD fonksiyonu ile bölünebilirlik kontrolü. AND operatörü her iki koşul da TRUE olmalı. Koşul sırası önemli: önce her ikisi, sonra tekler. FOR LOOP ve IF-ELSIF birlikte kullanımı. Bu tip sorular sınavlarda çok çıkar.

KONU 3: DML İşlemleri

INSERT, UPDATE, DELETE, MERGE

Soru 19 [KOLAY] - Basit INSERT

Departments tablosuna yeni bir departman ekleyin: ID=290, Name='Analytics', Location=1700.

ÇÖZÜM:

```
BEGIN
    INSERT INTO departments (department_id, department_name,
location_id)
    VALUES (290, 'Analytics', 1700);

    DBMS_OUTPUT.PUT_LINE('Yeni departman eklendi');
    COMMIT;
END;
```

AÇIKLAMA:

INSERT INTO tablo (kolonlar) VALUES (değerler) söz dizimi. Kolon sırası ve değer sırası eşleşmeli. String değerler tek tırnak içinde. COMMIT değişikliği kalıcı yapar. SQL%ROWCOUNT ile kaç satır etkilendi öğrenilebilir.

Soru 20 [KOLAY] - Basit UPDATE

Employee ID 150 olan çalışanın maaşını 5000$ yapın.

ÇÖZÜM:

```
BEGIN
    UPDATE employees
    SET salary = 5000
    WHERE employee_id = 150;

    DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' çalışan güncellendi');
    COMMIT;
END;
```

AÇIKLAMA:

UPDATE tablo SET kolon=değer WHERE koşul söz dizimi. WHERE çok önemli! WHERE olmazsa TÜM satırlar güncellenir. SQL%ROWCOUNT etkilenen satır sayısını döndürür. Primary key ile WHERE kullanınca kesin 1 satır etkilenir.

Soru 21 [KOLAY] - Basit DELETE

Job ID 'IT_PROG' olan tüm çalışanları silin.

ÇÖZÜM:

```
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM employees WHERE job_id =
'IT_PROG';
    DBMS_OUTPUT.PUT_LINE('Silinecek kayıt: ' || v_count);

    DELETE FROM employees WHERE job_id = 'IT_PROG';
```

```
    DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' kayıt silindi');
    COMMIT;
END;
```

AÇIKLAMA:

DELETE FROM tablo WHERE koşul söz dizimi. WHERE olmazsa TÜM satırlar silinir! Önce SELECT COUNT(*) ile kontrol etmek güvenli bir pratiktir. DELETE geri alınamaz (COMMIT sonrası). Önemli verilerde yedek alınmalı.

Soru 22 [ORTA] - Koşullu INSERT

Eğer salary 10000$'dan fazla olan çalışan sayısı 50'den azsa, yeni bir yüksek maaşlı pozisyon ekleyin.

ÇÖZÜM:

```
DECLARE
    v_high_salary_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_high_salary_count
    FROM employees WHERE salary > 10000;

    IF v_high_salary_count < 50 THEN
        INSERT INTO jobs (job_id, job_title, min_salary, max_salary)
        VALUES ('EX_MGR', 'Executive Manager', 15000, 30000);

        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Yeni pozisyon eklendi (Mevcut yüksek
maaşlı: ' || v_high_salary_count || ')');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Yeterli yüksek maaşlı çalışan var (' ||
v_high_salary_count || ')');
    END IF;
END;
```

AÇIKLAMA:

İş mantığı: önce kontrol et, sonra işlem yap. SELECT COUNT(*) toplam kayıt sayısı döndürür. IF ile koşullu DML yapılabilir. COMMIT sadece INSERT yapılırsa çalışır. İş kuralları böyle uygulanır.

Soru 23 [ORTA] - Toplu UPDATE

IT departmanındaki (dept_id=60) tüm çalışanların maaşına %12 zam yapın ve kaç kişinin etkilendiğini gösterin.

ÇÖZÜM:

```
DECLARE
    v_raise_percent NUMBER := 12;
    v_affected NUMBER;
BEGIN
    UPDATE employees
    SET salary = salary * (1 + v_raise_percent/100)
    WHERE department_id = 60;

    v_affected := SQL%ROWCOUNT;

    IF v_affected > 0 THEN
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(v_affected || ' IT çalışanına %' ||
v_raise_percent || ' zam yapıldı');
    ELSE
```

```
            DBMS_OUTPUT.PUT_LINE('IT departmanında çalışan bulunamadı');
        END IF;
    END;
```

AÇIKLAMA:

salary * (1 + 12/100) = salary * 1.12 yani %12 artış. Kendi üzerine işlem: salary = salary * ... SQL%ROWCOUNT hemen sonra okunmalı (başka SQL çalışmadan). Toplu güncelleme için tek UPDATE yeterli (LOOP gerekmez).

Soru 24 [ORTA] - INSERT SELECT

Maaşı 15000$'dan fazla olan çalışanları high_earners tablosuna kopyalayın.

ÇÖZÜM:

```
    BEGIN
        -- Önce tablo oluştur (sadece ilk çalıştırmada)
        -- CREATE TABLE high_earners AS SELECT * FROM employees WHERE
    1=0;

        INSERT INTO high_earners
        SELECT * FROM employees WHERE salary > 15000;

        DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' yüksek maaşlı çalışan
    kopyalandı');
        COMMIT;
    END;
```

AÇIKLAMA:

INSERT INTO tablo SELECT ... : sorgu sonucunu doğrudan tabloya ekler. VALUES yerine SELECT kullanılır. Toplu veri kopyalama için ideal. SELECT * tüm kolonları alır. WHERE koşulu ile filtreleme yapılır.

Soru 25 [ZOR] - MERGE İşlemi

Çalışan bonusları tablosunu güncelleyin: Kayıt varsa bonus güncelle (maaşın %10'u), yoksa yeni kayıt ekle.

ÇÖZÜM:

```
    BEGIN
        MERGE INTO employee_bonuses eb
        USING (SELECT employee_id, salary * 0.10 AS bonus_amt
               FROM employees
               WHERE department_id IN (50, 60, 80)) e
        ON (eb.employee_id = e.employee_id)
        WHEN MATCHED THEN
            UPDATE SET
                eb.bonus_amount = e.bonus_amt,
                eb.last_updated = SYSDATE
        WHEN NOT MATCHED THEN
            INSERT (employee_id, bonus_amount, last_updated)
            VALUES (e.employee_id, e.bonus_amt, SYSDATE);

        DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' kayıt işlendi
    (güncelleme/ekleme)');
        COMMIT;
    END;
```

AÇIKLAMA:

MERGE: UPSERT işlemi (UPDATE + INSERT). USING: kaynak veri tanımı. ON: eşleşme koşulu. WHEN MATCHED: kayıt varsa UPDATE. WHEN NOT MATCHED: kayıt yoksa INSERT. SYSDATE güncel tarih verir. Tek komutla her iki işlem.

Soru 26 [ZOR] - Karmaşık DELETE ve Yedekleme

5 yıldan eski ve maaşı düşük (<5000$) çalışanları silmeden önce archived_employees'a yedekleyin.

ÇÖZÜM:

```
DECLARE
    v_archive_count NUMBER;
    v_delete_count NUMBER;
BEGIN
    -- Önce yedekle
    INSERT INTO archived_employees
    SELECT *, SYSDATE AS archive_date
    FROM employees
    WHERE hire_date < ADD_MONTHS(SYSDATE, -60)  -- 5 yıl = 60 ay
    AND salary < 5000;

    v_archive_count := SQL%ROWCOUNT;

    -- Sonra sil
    DELETE FROM employees
    WHERE hire_date < ADD_MONTHS(SYSDATE, -60)
    AND salary < 5000;

    v_delete_count := SQL%ROWCOUNT;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Yedeklenen: ' || v_archive_count);
    DBMS_OUTPUT.PUT_LINE('Silinen: ' || v_delete_count);
END;
```

AÇIKLAMA:

Veri silmeden önce yedekleme önemli! ADD_MONTHS(tarih, -60) 60 ay (5 yıl) öncesini verir. INSERT SELECT * ile tüm kolonlar kopyalanır. SYSDATE AS archive_date yeni kolon ekler. İki işlem tek transaction'da (atomicity).

Soru 27 [SINAV] - Kapsamlı DML Senaryosu

Departman 50'deki çalışanlara ortalama maaşlarına göre zam yapın: Ortalamanın altındaysa %15, üstündeyse %5. Sonra toplam maaş bütçesini gösterin.

ÇÖZÜM:

```
DECLARE
    v_avg_salary NUMBER;
    v_below_avg_count NUMBER := 0;
    v_above_avg_count NUMBER := 0;
    v_total_budget NUMBER;
BEGIN
    -- Ortalama maaşı bul
    SELECT AVG(salary) INTO v_avg_salary
    FROM employees WHERE department_id = 50;

    DBMS_OUTPUT.PUT_LINE('Departman ortalama maaşı: $' ||
ROUND(v_avg_salary, 2));
```

```
    -- Ortalamanın altındakilere %15 zam
    UPDATE employees
    SET salary = salary * 1.15
    WHERE department_id = 50 AND salary < v_avg_salary;
    v_below_avg_count := SQL%ROWCOUNT;

    -- Ortalamanın üstündekilere %5 zam
    UPDATE employees
    SET salary = salary * 1.05
    WHERE department_id = 50 AND salary >= v_avg_salary;
    v_above_avg_count := SQL%ROWCOUNT;

    -- Yeni toplam bütçe
    SELECT SUM(salary) INTO v_total_budget
    FROM employees WHERE department_id = 50;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Altına %15 zam: ' || v_below_avg_count || '
kişi');
    DBMS_OUTPUT.PUT_LINE('Üstüne %5 zam: ' || v_above_avg_count || '
kişi');
    DBMS_OUTPUT.PUT_LINE('Yeni toplam bütçe: $' ||
ROUND(v_total_budget, 2));
END;
```

AÇIKLAMA:

Karmaşık iş senaryosu: önce analiz (AVG), sonra farklı UPDATE'ler, sonra toplam (SUM). AVG() ortalama, SUM() toplam hesaplar. Birden fazla UPDATE ayrı ayrı yapılır. Her UPDATE sonrası SQL%ROWCOUNT farklı değer döner. COMMIT en sonda tek sefer.

KONU 4: Transaction ve Veri Okuma

SELECT INTO, COMMIT, ROLLBACK, Transaction Yönetimi

Soru 28 [KOLAY] - Basit SELECT INTO

Employee ID 100 olan çalışanın adını ve maaşını çekin, ekrana yazdırın.

ÇÖZÜM:

```
DECLARE
    v_name VARCHAR2(50);
    v_salary NUMBER;
BEGIN
    SELECT first_name, salary
    INTO v_name, v_salary
    FROM employees
    WHERE employee_id = 100;

    DBMS_OUTPUT.PUT_LINE('Çalışan: ' || v_name);
    DBMS_OUTPUT.PUT_LINE('Maaş: $' || v_salary);
END;
```

AÇIKLAMA:

SELECT kolonlar INTO değişkenler FROM tablo WHERE koşul söz dizimi. Kolon sayısı = değişken sayısı olmalı. Sıralama önemli. TAM OLARAK BİR satır dönmeli. 0 satır = NO_DATA_FOUND, >1 satır = TOO_MANY_ROWS hatası.

Soru 29 [KOLAY] - COMMIT Kullanımı

Bir departmanın adını değiştirin ve kalıcı yapın.

ÇÖZÜM:

```
BEGIN
    UPDATE departments
    SET department_name = 'Information Technology'
    WHERE department_id = 60;

    DBMS_OUTPUT.PUT_LINE('Departman adı değiştirildi');
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Değişiklik kalıcı hale getirildi');
END;
```

AÇIKLAMA:

COMMIT değişiklikleri kalıcı yapar. COMMIT sonrası geri alınamaz. Transaction başlangıcından COMMIT'e kadar yapılan TÜM değişiklikler birlikte kaydedilir. DDL komutları (CREATE, DROP) otomatik COMMIT yapar.

Soru 30 [KOLAY] - ROLLBACK Kullanımı

Yanlışlıkla yapılan bir güncellemeyi geri alın.

ÇÖZÜM:

```
BEGIN
    UPDATE employees
    SET salary = salary * 2;  -- YANLIŞ! WHERE yok
```

```
    DBMS_OUTPUT.PUT_LINE('HATA tespit edildi!');
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('İşlem geri alındı, veriler eski haline
döndü');
END;
```

AÇIKLAMA:

ROLLBACK son COMMIT'ten bu yana yapılan TÜM değişiklikleri iptal eder. WHERE unutulmuş, tüm satırlar değişmiş. ROLLBACK ile kurtarılır. COMMIT yapılmışsa ROLLBACK çalışmaz! Session başında COMMIT yoktur, veritabanı başlangıç hali baseline'dır.

Soru 31 [ORTA] - Çoklu SELECT INTO

Bir departmanın tüm bilgilerini (name, manager, location) çekin ve formatla yazdırın.

ÇÖZÜM:

```
DECLARE
    v_dept_name departments.department_name%TYPE;
    v_manager_id departments.manager_id%TYPE;
    v_location_id departments.location_id%TYPE;
    v_manager_name VARCHAR2(100);
BEGIN
    SELECT department_name, manager_id, location_id
    INTO v_dept_name, v_manager_id, v_location_id
    FROM departments
    WHERE department_id = 50;

    -- Manager adını bul
    IF v_manager_id IS NOT NULL THEN
        SELECT first_name || ' ' || last_name INTO v_manager_name
        FROM employees WHERE employee_id = v_manager_id;
    ELSE
        v_manager_name := 'Atanmamış';
    END IF;

    DBMS_OUTPUT.PUT_LINE('Departman: ' || v_dept_name);
    DBMS_OUTPUT.PUT_LINE('Müdür: ' || v_manager_name);
    DBMS_OUTPUT.PUT_LINE('Lokasyon ID: ' || v_location_id);
END;
```

AÇIKLAMA:

%TYPE ile kolon tipi otomatik alınır. Birden fazla SELECT INTO yapılabilir. IS NULL / IS NOT NULL ile NULL kontrolü. NULL değerler özel kontrol gerektirir (= ile karşılaştırılamaz). İç içe SELECT INTO mantıklı veri akışı sağlar.

Soru 32 [ORTA] - SAVEPOINT Kullanımı

Üç adımlı işlem: INSERT, UPDATE, DELETE. UPDATE sonrası SAVEPOINT koy, DELETE hata verirse sadece DELETE'i geri al.

ÇÖZÜM:

```
DECLARE
    v_new_emp_id NUMBER := 999;
BEGIN
    -- Adım 1: INSERT
    INSERT INTO employees (employee_id, first_name, last_name, email,
hire_date, job_id)
```

```
    VALUES (v_new_emp_id, 'Test', 'User', 'TUSER', SYSDATE,
'IT_PROG');
    DBMS_OUTPUT.PUT_LINE('✓ INSERT tamamlandı');

    -- Adım 2: UPDATE
    UPDATE employees SET salary = 5000 WHERE employee_id =
v_new_emp_id;
    SAVEPOINT after_update;  -- Kayıt noktası
    DBMS_OUTPUT.PUT_LINE('✓ UPDATE tamamlandı, SAVEPOINT
oluşturuldu');

    -- Adım 3: DELETE (hata simülasyonu)
    BEGIN
        DELETE FROM employees WHERE employee_id = v_new_emp_id / 0;
-- Hata!
    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK TO after_update;  -- Sadece DELETE'i geri al
            DBMS_OUTPUT.PUT_LINE('✗ DELETE hatası, sadece DELETE geri
alındı');
    END;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('✓ INSERT ve UPDATE kaydedildi');
END;
```

AÇIKLAMA:

SAVEPOINT kayıt_noktası_adı: transaction içinde ara nokta. ROLLBACK TO kayıt_noktası: o noktaya geri döner, öncesini korur. COMMIT tüm savepoint'leri temizler. Karmaşık transaction'larda partial rollback için kullanılır. Bölüm / 0 hata verir (division by zero).

Soru 33 [ORTA] - EXCEPTION ile Transaction

Maaş güncellemesi yap, hata olursa ROLLBACK yap ve hata mesajı göster.

ÇÖZÜM:

```
DECLARE
    v_emp_id NUMBER := 100;
    v_new_salary NUMBER := 15000;
BEGIN
    UPDATE employees
    SET salary = v_new_salary
    WHERE employee_id = v_emp_id;

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Çalışan bulunamadı!');
    END IF;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Maaş güncellendi: $' || v_new_salary);

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('HATA: ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE('Tüm değişiklikler geri alındı');
END;
```

AÇIKLAMA:

EXCEPTION bloğu hataları yakalar. WHEN OTHERS tüm hataları yakalar.
RAISE_APPLICATION_ERROR(kod, mesaj) özel hata fırlatır (-20000 ile -20999 arası kullanıcı hataları).
SQLERRM hata mesajını döndürür. EXCEPTION'da ROLLBACK güvenli pratiktir.

Soru 34 [ZOR] - Atomik Para Transferi

Bir hesaptan diğerine para transferi: Her iki UPDATE da başarılı olmalı, değilse hiçbiri olmamalı.

ÇÖZÜM:

```
DECLARE
    v_from_emp NUMBER := 100;
    v_to_emp NUMBER := 101;
    v_amount NUMBER := 1000;
    v_from_balance NUMBER;
BEGIN
    -- Gönderen bakiyesini kontrol et
    SELECT salary INTO v_from_balance
    FROM employees WHERE employee_id = v_from_emp
    FOR UPDATE;  -- Satırı kilitle

    IF v_from_balance < v_amount THEN
        RAISE_APPLICATION_ERROR(-20001, 'Yetersiz bakiye!');
    END IF;

    -- Para çek
    UPDATE employees
    SET salary = salary - v_amount
    WHERE employee_id = v_from_emp;

    DBMS_OUTPUT.PUT_LINE('✓ ' || v_amount || '$ çekildi (ID: ' ||
v_from_emp || ')');

    -- Para yatır
    UPDATE employees
    SET salary = salary + v_amount
    WHERE employee_id = v_to_emp;

    DBMS_OUTPUT.PUT_LINE('✓ ' || v_amount || '$ yatırıldı (ID: ' ||
v_to_emp || ')');

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('✓✓ Transfer başarılı!');

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('✗✗ HATA: ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE('Transfer iptal edildi');
END;
```

AÇIKLAMA:

Atomicity: transaction ya tamamen olur ya hiç olmaz. FOR UPDATE satırı kilitler (başka session değiştiremez). Her iki UPDATE da başarılıysa COMMIT, hata varsa ROLLBACK. EXCEPTION bloğu güvenlik ağı. Gerçek bankacılık mantığı budur.

Soru 35 [ZOR] - Bulk Collection

Bir departmandaki tüm çalışanların maaşlarını bulk olarak çek, toplamını hesapla.

ÇÖZÜM:

```
DECLARE
    TYPE salary_table IS TABLE OF NUMBER;
    v_salaries salary_table;
    v_total NUMBER := 0;
    v_count NUMBER;
BEGIN
    SELECT salary
    BULK COLLECT INTO v_salaries
    FROM employees
    WHERE department_id = 50;

    v_count := v_salaries.COUNT;

    FOR i IN 1..v_count LOOP
        v_total := v_total + v_salaries(i);
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Çalışan sayısı: ' || v_count);
    DBMS_OUTPUT.PUT_LINE('Toplam maaş: $' || v_total);
    DBMS_OUTPUT.PUT_LINE('Ortalama maaş: $' || ROUND(v_total /
v_count, 2));
END;
```

AÇIKLAMA:

TYPE ... IS TABLE OF: collection (dizi) tanımlar. BULK COLLECT INTO: tüm satırları tek seferde collection'a atar (performanslı). collection.COUNT eleman sayısı verir. collection(index) ile erişilir (1'den başlar). FOR LOOP ile tüm elemanlar gezilir.

Soru 36 [SINAV] - Kapsamlı Transaction Senaryosu

Çalışanlara kıdemlerine göre bonus verin: 5+ yıl %20, 3-5 yıl %15, 1-3 yıl %10, <1 yıl bonus yok. Toplam bonus bütçesi 50000$ geçmesin. Geçerse tüm bonusları iptal et.

ÇÖZÜM:

```
DECLARE
    v_total_bonus NUMBER := 0;
    v_budget_limit NUMBER := 50000;
    v_years NUMBER;
    v_bonus NUMBER;
    v_count NUMBER := 0;
BEGIN
    FOR emp IN (SELECT employee_id, first_name, salary, hire_date
                FROM employees) LOOP
        -- Kıdem hesapla
        v_years := FLOOR(MONTHS_BETWEEN(SYSDATE, emp.hire_date) /
12);

        -- Bonus belirle
        IF v_years >= 5 THEN
            v_bonus := emp.salary * 0.20;
        ELSIF v_years >= 3 THEN
            v_bonus := emp.salary * 0.15;
        ELSIF v_years >= 1 THEN
            v_bonus := emp.salary * 0.10;
        ELSE
            v_bonus := 0;
        END IF;
```

```
        IF v_bonus > 0 THEN
            v_total_bonus := v_total_bonus + v_bonus;
            v_count := v_count + 1;

            -- Geçici bonus ekle
            UPDATE employees
            SET salary = salary + v_bonus
            WHERE employee_id = emp.employee_id;

            DBMS_OUTPUT.PUT_LINE(emp.first_name || ': ' || v_years ||
' yıl, $' || ROUND(v_bonus, 2));
        END IF;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('---');
    DBMS_OUTPUT.PUT_LINE('Toplam bonus: $' || ROUND(v_total_bonus,
2));
    DBMS_OUTPUT.PUT_LINE('Bütçe limiti: $' || v_budget_limit);

    -- Bütçe kontrolü
    IF v_total_bonus > v_budget_limit THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('✗ BÜTÇE AŞILDI! Tüm bonuslar iptal
edildi.');
        DBMS_OUTPUT.PUT_LINE('Eksik bütçe: $' || ROUND(v_total_bonus
- v_budget_limit, 2));
    ELSE
        COMMIT;
        DBMS_OUTPUT.PUT_LINE('✓ BONUSLAR ONAYLANDI! ' || v_count || '
kişiye bonus verildi.');
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('✗ HATA: ' || SQLERRM);
END;
```

AÇIKLAMA:

Karmaşık sınav senaryosu: FOR LOOP, IF-ELSIF, hesaplamalar, UPDATE, koşullu COMMIT/ROLLBACK, EXCEPTION. MONTHS_BETWEEN ile kıdem hesabı. Bütçe kontrolü iş mantığı. Tüm UPDATE'ler transaction içinde, bütçe aşılırsa hepsi iptal. Gerçek iş uygulaması simülasyonu.

□ Başarı İpuçları

Çalışma Stratejisi:

✓ Her soruyu kendi başınıza en az 2 kez yazın
✓ Kodları çalıştırın, sonuçları gözlemleyin
✓ Hata mesajlarını okuyun ve anlayın
✓ Farklı senaryolar deneyin (farklı ID'ler, değerler)
✓ WHERE koşulunu HER ZAMAN iki kez kontrol edin

Zorluk Seviyesi Açıklaması:

□ **KOLAY:** Temel söz dizimi, basit örnekler
□ **ORTA:** Birden fazla konsept birlikte, orta karmaşıklık
□ **ZOR:** İleri seviye, çoklu konsept, gerçek senaryolar
□ **SINAV:** Sınavda çıkabilecek tipik sorular

Kritik Hatalar:

✗ UPDATE/DELETE'te WHERE unutmak
✗ COMMIT yapmadan session kapatmak
✗ SELECT INTO'da birden fazla satır döndürmek
✗ ELSIF yerine ELSEIF yazmak
✗ Atama (:=) ile karşılaştırma (=) karıştırmak

36 Soru = 4 Konu × 9 Soru

Bol Bol Pratik Yapın! □

Sınavda Başarılar! □