Konu 1: Temel PL/SQL Blokları, Değişkenler ve `SELECT INTO`

Bu bölüm, PL/SQL programlamanın en temel yapı taşlarını anlamaya odaklanır: isimsiz bloklar, değişken tanımlama ve tek satırlık sorgu sonuçlarını değişkenlere atama.

Kolay Seviye

Soru 1.1 (Kolay): Ekrana "Merhaba PL/SQL!" yazdıran en basit PL/SQL bloğunu oluşturun.

Çözüm:

```SQL
BEGIN
  DBMS_OUTPUT.PUT_LINE("Merhaba PL/SQL!");
END;
/
```

Detaylı Açıklama:

- **Amaç:** En temel PL/SQL bloğunun yapısını göstermek.
- **Kod Açıklaması:** Her PL/SQL bloğu en azından bir `BEGIN` ve bir `END;` içermelidir. `BEGIN` yürütülebilir kodun başladığı, `END;` ise bittiği yeri belirtir. `DBMS_OUTPUT.PUT_LINE` Oracle'ın ekrana yazı yazdırmak için sunduğu standart bir prosedürdür. `/` işareti, SQL*Plus veya benzeri araçlarda bloğun bittiğini ve çalıştırılması gerektiğini belirtir.

Soru 1.2 (Kolay): `v_mesaj` adında bir `VARCHAR2(50)` değişkeni tanımlayın, içine "PL/SQL öğreniyorum." metnini atayın ve bu değişkenin içeriğini ekrana yazdırın.

Çözüm:

```SQL
DECLARE
  v_mesaj VARCHAR2(50) := 'PL/SQL öğreniyorum.';
BEGIN
  DBMS_OUTPUT.PUT_LINE(v_mesaj);
END;
/
```

Detaylı Açıklama:

- **Amaç:** Değişken tanımlama (`DECLARE`), değer atama (`:=`) ve değişken kullanma pratiği yapmak.
- **Kod Açıklaması:** `DECLARE` bölümü, programda kullanılacak değişkenlerin tanımlandığı yerdir. `v_mesaj` değişkenini 50 karakter uzunluğunda bir metin tutabilecek şekilde (`VARCHAR2(50)`) tanımladık. `:=` operatörü ile değişkene ilk değerini atadık. `BEGIN` bloğunda ise bu değişkeni `DBMS_OUTPUT.PUT_LINE` prosedürüne parametre olarak vererek içeriğini ekrana yazdırdık.

Soru 1.3 (Kolay): `employee_id` 'si 101 olan çalışanın adını (`first_name`) bir değişkene atayın ve ekrana yazdırın.

Çözüm:

SQL

```
DECLARE
  v_first_name employees.first_name%TYPE;
BEGIN
  SELECT first_name
  INTO v_first_name
  FROM employees
  WHERE employee_id = 101;

  DBMS_OUTPUT.PUT_LINE('101 ID''li çalışanın adı: ' || v_first_name);
END;
/
```

Detaylı Açıklama:

- **Amaç:** `SELECT INTO` kullanarak veritabanından tek bir değeri bir değişkene nasıl alacağımızı öğrenmek.
- **Kod Açıklaması:** `v_first_name` değişkenini `employees.first_name%TYPE` olarak tanımladık. Bu, değişkenin tipini `employees` tablosundaki `first_name` sütununun tipinden almasını sağlar, bu da kodumuzu daha esnek yapar. `SELECT first_name INTO v_first_name` ifadesi, sorgudan dönen `first_name` değerini `v_first_name` değişkenine atar. Metin içinde tek tırnak kullanmak için iki tane yan yana yazılır (`''`).

Orta Seviye

Soru 1.4 (Orta): `employee_id` 'si 102 olan çalışanın adını, soyadını ve maaşını üç ayrı değişkene atayıp "Çalışan: [Ad Soyad], Maaş: $[Maaş]" formatında ekrana yazdırın.

Çözüm:

SQL

```
DECLARE
  v_first_name employees.first_name%TYPE;
  v_last_name  employees.last_name%TYPE;
  v_salary     employees.salary%TYPE;
BEGIN
  SELECT first_name, last_name, salary
  INTO v_first_name, v_last_name, v_salary
  FROM employees
  WHERE employee_id = 102;

  DBMS_OUTPUT.PUT_LINE('Çalışan: ' || v_first_name || ' ' || v_last_name ||
', Maaş: $' || v_salary);
END;
/
```

Detaylı Açıklama:

- **Amaç:** `SELECT INTO` ile birden çok sütunu aynı anda birden çok değişkene atamayı öğrenmek.
- **Kod Açıklaması:** `SELECT` listesindeki sütunlar (`first_name` , `last_name` , `salary`) ile `INTO` listesindeki değişkenler (`v_first_name` , `v_last_name` , `v_salary`) hem sayı hem de sıra olarak eşleşmelidir. Oracle, birinci sütunu birinci değişkene, ikinci sütunu ikinciye vb. atar.

Soru 1.5 (Orta): `department_id` 'si 90 olan departmanın adını (`department_name`) `departments` tablosundan çekip ekrana yazdırın.

Çözüm:

SQL

```
DECLARE
  v_dept_name departments.department_name%TYPE;
  v_dept_id   departments.department_id%TYPE := 90;
BEGIN
  SELECT department_name
  INTO v_dept_name
  FROM departments
  WHERE department_id = v_dept_id;

  DBMS_OUTPUT.PUT_LINE(v_dept_id || ' numaralı departmanın adı: ' ||
v_dept_name);
END;
/
```

Detaylı Açıklama:

- **Amaç:** Farklı bir tablodan (`departments`) veri çekme ve `DECLARE` bloğunda değişkene başlangıç değeri atama pratiği yapmak.
- **Kod Açıklaması:** Bu örnekte, sorgulanacak `department_id` değerini `v_dept_id` adında bir değişkende saklıyoruz. Bu, kodun daha okunabilir olmasını sağlar ve eğer ID'yi birden çok yerde kullanmamız gerekseydi, sadece tek bir yerden değiştirmemiz yeterli olurdu.

Soru 1.6 (Orta): `job_id` 'si 'IT_PROG' olan işin minimum ve maksimum maaşını (`min_salary` , `max_salary`) `jobs` tablosundan iki ayrı değişkene atayın ve "İş Unvanı: IT_PROG, Maaş Aralığı: [min] - [max]" formatında yazdırın.

Çözüm:

SQL

```
DECLARE
  v_min_salary jobs.min_salary%TYPE;
  v_max_salary jobs.max_salary%TYPE;
  v_job_id     jobs.job_id%TYPE := 'IT_PROG';
BEGIN
  SELECT min_salary, max_salary
  INTO v_min_salary, v_max_salary
  FROM jobs
  WHERE job_id = v_job_id;

  DBMS_OUTPUT.PUT_LINE('İş Unvanı: ' || v_job_id || ', Maaş Aralığı: ' ||
v_min_salary || ' - ' || v_max_salary);
END;
/
```

Detaylı Açıklama:

- **Amaç:** `jobs` tablosu üzerinde çalışarak farklı bir senaryoda `SELECT INTO` kullanımını pekiştirmek.
- **Kod Açıklaması:** Mantık önceki sorularla aynıdır. `jobs` tablosundan, `job_id` 'si 'IT_PROG' olan kaydın `min_salary` ve `max_salary` değerleri çekilerek ilgili değişkenlere atanır ve formatlı bir şekilde ekrana yazdırılır.

Zor Seviye

Soru 1.7 (Zor): `employee_id` 'si 103 olan çalışanın tam adını (ad ve soyad birleşik), email adresini ve işe giriş tarihini `DD-MON-YYYY` formatında tek bir `VARCHAR2` değişkeninde birleştirip ekrana yazdırın.

Çözüm:

SQL

```
DECLARE
  v_emp_info VARCHAR2(200);
  v_emp_id   employees.employee_id%TYPE := 103;
BEGIN
  SELECT
    first_name || ' ' || last_name ||
    ' (Email: ' || email ||
    ', İşe Giriş: ' || TO_CHAR(hire_date, 'DD-MON-YYYY') || ')'
  INTO v_emp_info
  FROM employees
  WHERE employee_id = v_emp_id;

  DBMS_OUTPUT.PUT_LINE('Çalışan Bilgisi: ' || v_emp_info);
END;
/
```

Detaylı Açıklama:

- **Amaç:** SQL sorgusu içinde metin birleştirme (`||`) ve veri tipi dönüşüm fonksiyonlarını (`TO_CHAR`) kullanarak tek bir sonuç üretip bunu PL/SQL değişkenine almayı göstermek.
- **Kod Açıklaması:** Bu örnekte, tüm formatlama ve birleştirme işlemini `SELECT` ifadesinin içinde yapıyoruz. `first_name` ve `last_name` birleştiriliyor, `email` ekleniyor ve en önemlisi `hire_date` (bir `DATE` veri tipi) `TO_CHAR` fonksiyonu ile istediğimiz metin formatına (`DD-MON-YYYY`) dönüştürülüyor. Sorgunun ürettiği bu tek metin sonucu, `v_emp_info` değişkenine atanır.

Soru 1.8 (Zor): `employee_id` 'si 176 olan çalışanın yöneticisinin (`manager_id`) kim olduğunu bulun. Ardından, bulduğunuz yönetici ID'si ile `employees` tablosundan yöneticinin adını ve soyadını başka bir sorgu ile bularak "Çalışan 176''nın yöneticisi: [Yönetici Adı Soyadı]" şeklinde yazdırın.

Çözüm:

SQL

```
DECLARE
  v_manager_id    employees.manager_id%TYPE;
  v_manager_name  VARCHAR2(100);
  v_employee_id   employees.employee_id%TYPE := 176;
BEGIN
  -- 1. Adım: Çalışanın yönetici ID'sini bul
  SELECT manager_id
```

```
  INTO v_manager_id
  FROM employees
  WHERE employee_id = v_employee_id;

  -- 2. Adım: Bulunan yönetici ID'si ile yöneticinin adını ve soyadını bul
  SELECT first_name || ' ' || last_name
  INTO v_manager_name
  FROM employees
  WHERE employee_id = v_manager_id;

  DBMS_OUTPUT.PUT_LINE('Çalışan ' || v_employee_id || '''nın yöneticisi: '
|| v_manager_name);
END;
/
```

Detaylı Açıklama:

- **Amaç:** Bir sorgudan elde edilen sonucu, başka bir sorgu için girdi olarak kullanmayı göstererek adım adım problem çözme mantığını geliştirmek.
- **Kod Açıklaması:** Bu, iki adımlı bir işlemdir. İlk `SELECT INTO` sorgusu, 176 ID'li çalışanın `manager_id`'sini alır ve `v_manager_id` değişkenine atar. İkinci `SELECT INTO` sorgusu ise bu `v_manager_id` değişkenini `WHERE` koşulunda kullanarak yöneticinin bilgilerini çeker. Bu, PL/SQL'in prosedürel gücünü gösterir; SQL'de tek sorguda yapılması zor olabilecek bir işlemi, adımlara bölerek kolayca çözebiliriz.

Sınav Sorusu

Soru 1.9 (Sınav): Bir çalışanın ID'sini bir değişkene atayın (örneğin, 145). Bu çalışanın maaşını ve çalıştığı departmanın adını iki ayrı değişkene atayın. Eğer çalışanın maaşı 10000'den büyükse **VE** departman adı 'Sales' ise ekrana "Satış departmanında kıdemli çalışan." yazdırın. Aksi takdirde "Diğer" yazdırın. Bu problemi, tabloları `JOIN` kullanmadan, iki ayrı `SELECT INTO` sorgusu ile çözün.

Çözüm:

SQL

```
DECLARE
  v_employee_id   employees.employee_id%TYPE := 145;
  v_salary        employees.salary%TYPE;
  v_department_id employees.department_id%TYPE;
  v_dept_name     departments.department_name%TYPE;
BEGIN
  -- 1. Adım: Çalışanın maaşını ve departman ID'sini al
  SELECT salary, department_id
  INTO v_salary, v_department_id
```

```
  FROM employees
  WHERE employee_id = v_employee_id;

  -- 2. Adım: Alınan departman ID'si ile departman adını bul
  SELECT department_name
  INTO v_dept_name
  FROM departments
  WHERE department_id = v_department_id;

  -- 3. Adım: Koşulları kontrol et
  IF v_salary > 10000 AND v_dept_name = 'Sales' THEN
    DBMS_OUTPUT.PUT_LINE('Satış departmanında kıdemli çalışan.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Diğer');
  END IF;
END;
/
```

Detaylı Açıklama:

- **Amaç:** Bu soru, bir sınavda adayın birden çok temel yeteneğini ölçmeyi hedefler: değişken yönetimi, birden çok tablodan adım adım veri çekme ve `IF` bloğu içinde birden çok koşulu (`AND` ile) birleştirme.
- **Kod Açıklaması:** Önce `employees` tablosundan maaş ve departman ID'si alınır. Sonra bu departman ID'si kullanılarak `departments` tablosundan departman adı bulunur. En son adımda ise `IF` bloğu, hem maaş (`v_salary > 10000`) hem de departman adı (`v_dept_name = 'Sales'`) koşullarının ikisinin de aynı anda doğru olup olmadığını `AND` operatörü ile kontrol eder. Sadece iki koşul da sağlanırsa `THEN` bloğu çalışır. Bu, gerçek hayattaki iş kurallarının nasıl koda döküldüğüne dair temel bir örnektir.

Konu 2: Koşullu Kontrol (`IF` ve `CASE`)

Bu bölüm, programın akışını belirli koşullara göre yönlendiren `IF` ve `CASE` yapılarının kullanımını pekiştirmeyi amaçlar.

Kolay Seviye

Soru 2.1 (Kolay): Bir sayısal değişken tanımlayın ve değerini 25 olarak atayın. Eğer değişkenin değeri 18'den büyükse ekrana "Reşit" yazdırın.

Çözüm:

SQL

```
DECLARE
  v_yas NUMBER := 25;
BEGIN
  IF v_yas > 18 THEN
    DBMS_OUTPUT.PUT_LINE("Reşit");
  END IF;
END;
/
```

Detaylı Açıklama:

- **Amaç:** En basit `IF-THEN-END IF` yapısını anlamak.
- **Kod Açıklaması:** `IF` anahtar kelimesinden sonra gelen koşul (`v_yas > 18`) değerlendirilir. Sonuç `TRUE` ise, `THEN` ve `END IF` arasındaki kod çalıştırılır. Koşul `FALSE` veya `NULL` ise, bu blok atlanır.

Soru 2.2 (Kolay): `employee_id` 'si 110 olan çalışanın `job_id` 'sini kontrol edin. Eğer `job_id` 'si 'AC_ACCOUNT' ise ekrana "Muhasebe çalışanı" yazdırın, değilse "Diğer departman" yazdırın.

Çözüm:

SQL

```
DECLARE
  v_job_id employees.job_id%TYPE;
BEGIN
  SELECT job_id
  INTO v_job_id
  FROM employees
  WHERE employee_id = 110;

  IF v_job_id = 'AC_ACCOUNT' THEN
    DBMS_OUTPUT.PUT_LINE('Muhasebe çalışanı');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Diğer departman');
  END IF;
END;
/
```

Detaylı Açıklama:

- **Amaç:** `IF-THEN-ELSE` yapısını kullanarak bir koşulun hem doğru hem de yanlış olması durumlarını yönetmek.
- **Kod Açıklaması:** `IF` koşulu `TRUE` ise `THEN` bloğu çalışır. Koşul `FALSE` ise, program akışı `ELSE` anahtar kelimesinden sonraki bloğa geçer ve oradaki kodu çalıştırır. Bu, her

durumda bir eylem gerçekleştirilmesini sağlar.

Soru 2.3 (Kolay): Bir `BOOLEAN` değişkeni tanımlayın ve değerini `TRUE` olarak atayın. Bu değişkenin değerine göre ekrana "Doğru" veya "Yanlış" yazdırın.

Çözüm:

SQL

```
DECLARE
  v_dogru_mu BOOLEAN := TRUE;
BEGIN
  IF v_dogru_mu THEN
    DBMS_OUTPUT.PUT_LINE('Doğru');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Yanlış');
  END IF;
END;
/
```

Detaylı Açıklama:

- **Amaç:** `BOOLEAN` veri tipinin `IF` koşullarında doğrudan nasıl kullanıldığını göstermek.
- **Kod Açıklaması:** Değişkenin kendisi zaten `TRUE` veya `FALSE` olduğu için, `IF v_dogru_mu = TRUE THEN` yazmaya gerek yoktur. `IF v_dogru_mu THEN` ifadesi, değişkenin `TRUE` olup olmadığını kontrol etmek için yeterlidir.

Orta Seviye

Soru 2.4 (Orta): `employee_id`'si 104 olan çalışanın maaşını kontrol edin. Maaşı 5000'den az ise "%10 zam önerildi", 5000 ile 10000 arasında ise "%5 zam önerildi", 10000'den fazla ise "Zam önerilmedi" yazdırın.

Çözüm:

SQL

```
DECLARE
  v_salary employees.salary%TYPE;
BEGIN
  SELECT salary
  INTO v_salary
  FROM employees
  WHERE employee_id = 104;

  IF v_salary < 5000 THEN
    DBMS_OUTPUT.PUT_LINE('%10 zam önerildi');
```

```
  ELSIF v_salary BETWEEN 5000 AND 10000 THEN
    DBMS_OUTPUT.PUT_LINE('%5 zam önerildi');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Zam önerilmedi');
  END IF;
END;
/
```

Detaylı Açıklama:

- **Amaç:** `IF-THEN-ELSIF-ELSE` yapısını kullanarak ikiden fazla durumu yönetmek.
- **Kod Açıklaması:** `ELSIF` (ELSE IF'in kısaltması), bir önceki `IF` veya `ELSIF` koşulu `FALSE` olduğunda yeni bir koşulun test edilmesini sağlar. PL/SQL, koşulları yukarıdan aşağıya doğru test eder ve `TRUE` olan ilk koşulu bulduğunda ilgili bloğu çalıştırır ve `IF` yapısından çıkar. `BETWEEN ... AND ...` operatörü, bir değerin belirli bir aralıkta olup olmadığını kontrol etmek için kullanılır (sınırlar dahildir).

Soru 2.5 (Orta): `department_id`'si 50 olan departmanın `location_id`'sini `departments` tablosundan bulun. `location_id` 1500 ise "Seattle", 1700 ise "South San Francisco", diğer durumlarda ise "Bilinmeyen Lokasyon" yazdıran bir `Simple CASE` yapısı kurun.

Çözüm:

SQL

```
DECLARE
  v_location_id departments.location_id%TYPE;
BEGIN
  SELECT location_id
  INTO v_location_id
  FROM departments
  WHERE department_id = 50;

  CASE v_location_id
    WHEN 1500 THEN
      DBMS_OUTPUT.PUT_LINE('Seattle');
    WHEN 1700 THEN
      DBMS_OUTPUT.PUT_LINE('South San Francisco');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Bilinmeyen Lokasyon');
  END CASE;
END;
/
```

Detaylı Açıklama:

- **Amaç:** `Simple CASE` yapısının, tek bir değişkenin değerini birden çok sabit değerle karşılaştırmak için nasıl kullanıldığını göstermek.
- **Kod Açıklaması:** `CASE v_location_id` ifadesi, `v_location_id` değişkenini seçici olarak belirler. `WHEN` dalları, bu değişkenin değerini sırasıyla `1500` ve `1700` ile karşılaştırır. Eşleşen ilk `WHEN` bloğu çalıştırılır ve `CASE` yapısından çıkılır. Hiçbir `WHEN` eşleşmezse, `ELSE` bloğu çalışır. `Simple CASE`, bu tür senaryolar için `IF-ELSIF`'e göre daha okunabilir bir alternatiftir.

Soru 2.6 (Orta): `employee_id`'si 178 olan çalışanın `job_id` ve `salary` bilgilerini alın. `job_id`'si 'SA_REP' **VE** maaşı 8000'den büyükse "Kıdemli Satış Temsilcisi", `job_id`'si 'SA_REP' **VE** maaşı 8000'den küçük veya eşitse "Satış Temsilcisi" yazdırın.

Çözüm:

SQL

```
DECLARE
  v_job_id employees.job_id%TYPE;
  v_salary employees.salary%TYPE;
BEGIN
  SELECT job_id, salary
  INTO v_job_id, v_salary
  FROM employees
  WHERE employee_id = 178;

  IF v_job_id = 'SA_REP' AND v_salary > 8000 THEN
    DBMS_OUTPUT.PUT_LINE('Kıdemli Satış Temsilcisi');
  ELSIF v_job_id = 'SA_REP' AND v_salary <= 8000 THEN
    DBMS_OUTPUT.PUT_LINE('Satış Temsilcisi');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Satış dışı personel');
  END IF;
END;
/
```

Detaylı Açıklama:

- **Amaç:** `IF` koşulları içinde `AND` mantıksal operatörünü kullanarak birden çok koşulu birleştirmeyi göstermek.
- **Kod Açıklaması:** `AND` operatörü, her iki tarafındaki koşulun da `TRUE` olması durumunda `TRUE` sonucunu üretir. Bu örnekte, çalışanın hem `job_id`'si hem de maaşı aynı anda kontrol edilerek daha spesifik bir karar verilir.

Zor Seviye

Soru 2.7 (Zor): `employee_id` 'si 145 olan çalışanın `commission_pct` (komisyon oranı) değerini kontrol edin. Eğer komisyon oranı `NULL` ise "Komisyon almıyor", 0.1 ile 0.2 arasında ise "Düşük Komisyon", 0.21 ile 0.3 arasında ise "Orta Komisyon", 0.3'ten büyükse "Yüksek Komisyon" yazdıran bir `Searched CASE` yapısı kurun.

Çözüm:

SQL

```
DECLARE
  v_commission_pct employees.commission_pct%TYPE;
BEGIN
  SELECT commission_pct
  INTO v_commission_pct
  FROM employees
  WHERE employee_id = 145;

  CASE
    WHEN v_commission_pct IS NULL THEN
      DBMS_OUTPUT.PUT_LINE('Komisyon almıyor');
    WHEN v_commission_pct BETWEEN 0.1 AND 0.2 THEN
      DBMS_OUTPUT.PUT_LINE('Düşük Komisyon');
    WHEN v_commission_pct BETWEEN 0.21 AND 0.3 THEN
      DBMS_OUTPUT.PUT_LINE('Orta Komisyon');
    WHEN v_commission_pct > 0.3 THEN
      DBMS_OUTPUT.PUT_LINE('Yüksek Komisyon');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Tanımlanmamış Komisyon Aralığı');
  END CASE;
END;
/
```

Detaylı Açıklama:

- **Amaç:** `Searched CASE` yapısının, her `WHEN` dalında farklı ve karmaşık bir koşulun (örneğin `IS NULL`, `BETWEEN`) nasıl değerlendirilebildiğini göstermek.
- **Kod Açıklaması:** `Searched CASE` 'de, `CASE` anahtar kelimesinden sonra bir seçici değişken yoktur. Her `WHEN` kendi başına tam bir koşul içerir. Bu, `Simple CASE` 'in yetersiz kaldığı, farklı türde karşılaştırmalar (eşitlik, büyüklük, `NULL` kontrolü vb.) yapılması gereken durumlar için idealdir. `IS NULL` operatörü, bir değerin `NULL` olup olmadığını kontrol eder; `=` operatörü `NULL` ile kullanılamaz.

Soru 2.8 (Zor): `employee_id` 'si 205 olan çalışanın `hire_date` (işe giriş tarihi) bilgisini alın. Eğer çalışan 2005 yılından önce işe girdiyse **VEYA** `job_id` 'si 'MK_MAN' ise ekrana "Kıdemli

Pazarlamacı" yazdırın. Aksi takdirde, sadece 2005 yılından sonra işe girdiyse "Yeni Pazarlamacı" yazdırın.

Çözüm:

SQL

```
DECLARE
  v_hire_date employees.hire_date%TYPE;
  v_job_id    employees.job_id%TYPE;
BEGIN
  SELECT hire_date, job_id
  INTO v_hire_date, v_job_id
  FROM employees
  WHERE employee_id = 205;

  IF TO_CHAR(v_hire_date, 'YYYY') < '2005' OR v_job_id = 'MK_MAN' THEN
    DBMS_OUTPUT.PUT_LINE('Kıdemli Pazarlamacı');
  ELSIF TO_CHAR(v_hire_date, 'YYYY') >= '2005' THEN
    DBMS_OUTPUT.PUT_LINE('Yeni Pazarlamacı');
  END IF;
END;
/
```

Detaylı Açıklama:

- **Amaç:** `OR` mantıksal operatörünü ve iç içe mantığı kullanarak karmaşık bir iş kuralını uygulamak.
- **Kod Açıklaması:** `OR` operatörü, iki koşuldan en az birinin `TRUE` olması durumunda `TRUE` sonucunu üretir. İlk `IF` koşulu, çalışanın ya 2005'ten önce işe girmiş olmasını ya da unvanının 'MK_MAN' olmasını kontrol eder. Bu iki durumdan herhangi biri doğruysa, "Kıdemli Pazarlamacı" mesajı yazdırılır. `TO_CHAR(v_hire_date, 'YYYY')` fonksiyonu, `DATE` tipindeki tarihten sadece yıl bilgisini metin olarak alır ve karşılaştırma yapılmasını sağlar.

Sınav Sorusu

Soru 2.9 (Sınav): `employee_id` 'si 174 olan çalışanın maaşını, komisyon oranını ve `job_id` 'sini alın. Aşağıdaki iç içe geçmiş kurallara göre çalışanın gelir durumunu belirleyin:

1. Öncelikle, çalışanın `job_id` 'sinin 'SA_REP' olup olmadığını kontrol edin.
2. Eğer 'SA_REP' ise, komisyon oranının (`commission_pct`) `NULL` olup olmadığını kontrol edin.

- Eğer komisyon oranı `NULL` değilse, ekrana "Komisyonlu Satış Temsilcisi" yazdırın.

- Eğer komisyon oranı `NULL` ise, ekrana "Sabit Maaşlı Satış Temsilcisi" yazdırın.

1. Eğer 'SA_REP' değilse, maaşının 10000'den büyük olup olmadığını kontrol edin.

- Eğer maaşı 10000'den büyükse, ekrana "Yüksek Maaşlı Yönetici/Uzman" yazdırın.
- Değilse, ekrana "Standart Maaşlı Personel" yazdırın.

Çözüm:

SQL

```
DECLARE
  v_salary         employees.salary%TYPE;
  v_commission_pct employees.commission_pct%TYPE;
  v_job_id         employees.job_id%TYPE;
BEGIN
  SELECT salary, commission_pct, job_id
  INTO v_salary, v_commission_pct, v_job_id
  FROM employees
  WHERE employee_id = 174;

  IF v_job_id = 'SA_REP' THEN
    -- İç IF: Sadece job_id 'SA_REP' ise bu blok çalışır
    IF v_commission_pct IS NOT NULL THEN
      DBMS_OUTPUT.PUT_LINE('Komisyonlu Satış Temsilcisi');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Sabit Maaşlı Satış Temsilcisi');
    END IF;
  ELSE
    -- İç IF: job_id 'SA_REP' değilse bu blok çalışır
    IF v_salary > 10000 THEN
      DBMS_OUTPUT.PUT_LINE('Yüksek Maaşlı Yönetici/Uzman');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Standart Maaşlı Personel');
    END IF;
  END IF;
END;
/
```

Detaylı Açıklama:

- **Amaç:** Bu soru, adayın **iç içe** `IF` **(nested** `IF` **)** yapılarını kurma ve karmaşık, hiyerarşik iş kurallarını koda dökme becerisini ölçer.
- **Kod Açıklaması:** Ana `IF` yapısı, çalışanın `job_id` 'sine göre programı iki ana dala ayırır. Her bir dalın içinde, o dala özgü ek bir koşulu kontrol eden ikinci bir `IF` yapısı bulunur. Bu, bir koşulun sonucuna göre başka bir koşulu değerlendirme mantığıdır. Örneğin, bir kişinin satışçı olup olmadığını bilmeden komisyonunu sormak anlamsızdır. Bu kod, tam

olarak bu hiyerarşiyi yansıtır: önce unvanı kontrol et, sonra unvana göre ilgili detayı (komisyon veya maaş) kontrol et. Bu yapı, karmaşık karar ağaçlarını modellemek için çok güçlüdür.