```
-------------------------------------------
--Output Log Yazdırma
-------------------------------------------

set serveroutput on
begin

  dbms_output.put_line('Hello World');

end;

-------------------------------------------

set serveroutput on
declare
    p_maas number;

begin

    select maas into p_maas from personel
    where personel_id = 5006;

    dbms_output.put_line(p_maas);

end;
```

```
-------------------------------------------
--Değişkenlere Değer Atama
-------------------------------------------

declare
    v_sehir varchar2(50);
    v_ulke  varchar2(50) := 'Türkiye';

begin

    dbms_output.put_line('Şehir İsmi: ' || v_sehir);

    v_sehir := 'İstanbul';

    dbms_output.put_line('Şehir İsmi: ' || v_sehir);

    dbms_output.put_line('-----------------------------');

    v_ulke := 'Almanya';

    dbms_output.put_line('Ülke İsmi: ' || v_ulke);

end;

-------------------------------------------
--Değişkenleri SQL İçinde Kullanma
-------------------------------------------

declare
    v_maas number;
    v_ad varchar2(50);

begin

    select maas, ad into v_maas, v_ad
    from personel
    where personel_id = 5006;

    dbms_output.put_line('Adı: '||v_ad||', Maaşı:'||v_maas);

end;

-------------------------------------------
--String Veri Tipleri
-------------------------------------------

declare
    v_char    char(10);
    v_varchar varchar2(10);
    v_clob    clob;
begin
    v_char    := 'Oracle';
    v_varchar := 'Oracle';
    v_clob    := 'Oracle';

    dbms_output.put_line(v_char);
    dbms_output.put_line(v_varchar);
    dbms_output.put_line(v_clob);

end;
```

```
-------------------------------------------
-- String Ayıracı
-------------------------------------------

declare
    v_etkinlik varchar2(20);

begin

    v_etkinlik := 'Ahmet''in doğum günü';
    dbms_output.put_line(v_etkinlik);

    v_etkinlik := q'!Selen'in partisi!';
    dbms_output.put_line(v_etkinlik);

    v_etkinlik := q'[BMW'nin tanıtımı]';
    dbms_output.put_line(v_etkinlik);

end;

-------------------------------------------
--Tarih-Zaman Veri Tipleri
-------------------------------------------

declare
  v_sure1 interval year to month := '03-02';
  v_sure2 interval year to month := interval '6' month;
  v_sure3 interval day to second := '40 10:20:10';
  v_sure4 interval day to second := interval '5 5:5:5' day to second;

begin

  dbms_output.put_line(to_char(sysdate, 'dd.mm.yyyy hh24:mi:ss'));
  dbms_output.put_line(sysdate + v_sure1);
  dbms_output.put_line(sysdate + v_sure2);
  dbms_output.put_line(to_char(sysdate + v_sure3, 'dd.mm.yyyy hh24:mi:ss'));
  dbms_output.put_line(to_char(sysdate + v_sure4, 'dd.mm.yyyy hh24:mi:ss'));

end;

-------------------------------------------
--Veri Tiplerini Dönüştürme
-------------------------------------------

declare
  v_maas varchar2(10) := '10000';
  v_prim number := 2000;
  v_toplam number;

begin

  v_toplam := v_maas + v_prim;
  dbms_output.put_line(v_toplam);

end;

-------------------------------------------

declare
  v_maas varchar2(10) := '10000';
  v_prim number := 2000;
  v_toplam number;

begin
  v_toplam := to_number(v_maas) + v_prim;
  dbms_output.put_line(v_toplam);
end;
```

```
-------------------------------------------

declare
    v_tarih1 date;
    v_tarih2 date;
    v_tarih3 date;

begin

    v_tarih1 := to_date('15 February 2020', 'dd month yyyy');
    v_tarih2 := '15.02.2020';
    v_tarih3 := '15 February 2020';

    dbms_output.put_line(v_tarih1);
    dbms_output.put_line(v_tarih2);
    dbms_output.put_line(v_tarih3);

end;

-------------------------------------------
--%TYPE İle Değişken Tanımlama
-------------------------------------------

declare
    v_maas  personel.maas%type;
    v_ad    personel.ad%type;
    v_tarih personel.giris_tarihi%type;

begin

    select maas, ad, giris_tarihi
    into   v_maas, v_ad, v_tarih
    from personel
    where personel_id = 5010;

    dbms_output.put_line('Adı: '   ||v_ad);
    dbms_output.put_line('Maaşı:'  ||v_maas);
    dbms_output.put_line('Tarihi:' ||v_tarih);

end;

-------------------------------------------
--Boolean Değişken Tanımlama
-------------------------------------------

declare
    v_uygun         boolean;
    v_tarih1        date := to_date('01.02.2021', 'dd.mm.yyyy');
    v_tarih2        date;

begin

    v_tarih2 := to_date('01 February 2021', 'dd month yyyy');

    if v_tarih1 = v_tarih2 then
       v_uygun := true;
    else
       v_uygun := false;
    end if;

    if v_uygun then
       dbms_output.put_line('TRUE – Tarihler Aynı');
    else
        dbms_output.put_line('FALSE – Tarihler Farklı');
    end if;

end;
```

```
-------------------------------------------

declare
    v_pers_maas number;
    v_max_maas number := 5000;
    v_maas_uygun boolean := false;
    v_personel_id pls_integer := 5005;

begin
    select maas into v_pers_maas
    from personel
    where personel_id = v_personel_id;

    if v_pers_maas < v_max_maas then
       v_maas_uygun := true;
    end if;

    if v_maas_uygun then
       dbms_output.put_line('Maaş uygun');
    else
       dbms_output.put_line('Maaş uygun değil: '|| v_pers_maas);
    end if;
end;

-------------------------------------------
--Bind Değişkenler
-------------------------------------------

variable b_sonuc number
begin

    select sum(maas) into :b_sonuc from personel;

end;
print b_sonuc;

-------------------------------------------

variable b_sonuc number
set autoprint on
declare
  v_unvan varchar2(30) := &unvan;
begin

    select sum(maas) into :b_sonuc
    from personel where unvan = v_unvan;

end;

-------------------------------------------

variable b_unvan varchar2(30)
begin

    select unvan into :b_unvan from personel where personel_id = 5000;

end;

print b_unvan;
select ad, soyad, unvan from personel where unvan = :b_unvan;
```

```
-------------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------------

-----------------------------------------
--Değişkenleri SQL İçinde Kullanma
-----------------------------------------

declare
    v_name varchar2(50);
    v_title varchar2(150);
    v_title_of_courtesy varchar2(50);
begin

    select e.first_name || ' ' || e.last_name, e.title, e.title_of_courtesy
        into v_name, v_title, v_title_of_courtesy
    from employees e
    where e.employee_id = 7;

    dbms_output.put_line(v_title_of_courtesy || ' ' || v_name||', '||v_title);

end;

-----------------------------------------
--String Veri Tipleri
-----------------------------------------

declare
    v_str1 varchar2(20);
    v_str2 varchar2(20);
    v_str3 varchar2(20);

begin

    v_str1    := 'Ankara';
    v_str2    := 'İzmir';
    v_str3    := v_str1 || ',' || v_str2;

    dbms_output.put_line(v_str3);

end;

-----------------------------------------
--Tarih-Zaman Veri Tipleri
-----------------------------------------

declare
    v_tarih1 date;
    v_sure interval day to second := '40 01:00:00';

begin

    v_tarih1 := to_date('01.09.' || to_char(sysdate,'yyyy') || ' 19:00', 'dd.mm.yyyy hh24:mi');

    dbms_output.put_line('1. Seminer Tarihi: ' || to_char(v_tarih1, 'dd.mm.yyyy hh24:mi:ss'));
    dbms_output.put_line('2. Seminer Tarihi: ' || to_char(v_tarih1 + v_sure, 'dd.mm.yyyy
hh24:mi:ss'));
    dbms_output.put_line('3. Seminer Tarihi: ' || to_char(v_tarih1 + v_sure + v_sure, 'dd.mm.yyyy
hh24:mi:ss'));

end;
```

```
--Alternatif cevap:

select
    to_char(tarih1,'dd.mm.yyyy hh24:mi') tarih1,
    to_char(tarih1+sure,'dd.mm.yyyy hh24:mi') tarih2,
    to_char(tarih1+sure*2,'dd.mm.yyyy hh24:mi') tarih3
from
(
    select to_date('01.09.'||to_char(sysdate,'yyyy')||' 19:00', 'dd.mm.yyyy hh24:mi') tarih1,
        to_dsinterval('40 01:00:00') sure
    from dual
)

-------------------------------------------
--%TYPE İle Değişken Tanımlama
-------------------------------------------

declare
    v_customer_id   customers.customer_id%type;
    v_company_name  customers.company_name%type;
    v_order_date    orders.order_date%type;
    v_freight       orders.freight%type;
begin

    select c.customer_id, c.company_name, o.order_date, o.freight
        into v_customer_id, v_company_name, v_order_date, v_freight
    from orders o, customers c
    where order_id = 10303
        and o.customer_id = c.customer_id;

    dbms_output.put_line('Customer ID: '|| v_customer_id);
    dbms_output.put_line('Company Name: '|| v_company_name);
    dbms_output.put_line('Order Date: '|| v_order_date);
    dbms_output.put_line('Order Amount: '|| v_freight);

end;

-------------------------------------------
--Boolean Değişken Tanımlama
-------------------------------------------

declare
    v_hedef integer := 90;
    v_hedef_sonucu boolean := false;
    v_count smallint;
begin

    select count(*) into v_count from customers;

    if v_count > v_hedef then
        v_hedef_sonucu := true;
    end if;

    if v_hedef_sonucu then
        dbms_output.put_line('Müşteri hedefine ulaşıldı, tebrikler :)');
    else
        dbms_output.put_line('Biraz daha gayret lütfen!');
    end if;

end;
```

```
-------------------------------------------
--Bind Değişkenler
-------------------------------------------

variable v_sum_freight_koln number
set autoprint on

begin

    select sum(freight) into :v_sum_freight_koln
    from orders
    where ship_city = 'Köln';

end;
/

select ship_city, sum(freight) from orders
group by ship_city
having sum(freight) > :v_sum_freight_koln;
```

```
-------------------------------------------
--PL/SQL İçinde Kullanılan SQL Fonksiyonları
-------------------------------------------

declare
  v_deger number;
  v_str  varchar2(50) := 'PL/SQL eğitimi için doğru yerdesiniz';

begin

  v_deger := length(v_str);
  v_deger := months_between(baslangic_tarihi, bitis_tarihi);

end;

-------------------------------------------

declare
  v_deger number := 1234.567;
  v_str   varchar2(50) := 'PL/SQL eğitimi için doğru yerdesiniz.';

begin

  dbms_output.put_line(instr(v_str,'SQL'));
  dbms_output.put_line(concat('Tuncay ', 'Tiryaki'));
  dbms_output.put_line(to_char(v_deger));
  dbms_output.put_line(lower(v_str));
  dbms_output.put_line(substr(v_str, 8, 7));
  dbms_output.put_line(replace(v_str, 'doğru', 'en doğru'));
  dbms_output.put_line(round(v_deger,2));
  dbms_output.put_line(last_day(sysdate));

end;

-------------------------------------------
--Sequence Kullanma
-------------------------------------------

create sequence sq_temp
start with 1
increment by 1;

declare
  v_seq_number number;
begin

  v_seq_number := sq_temp.nextval;
  dbms_output.put_line('Sıra Numarası:' || v_seq_number);

end;

begin

  insert into konum
  values (sq_temp.nextval, 'Yeni Konum', 12);

end;
```

```
-------------------------------------------
--Nested (İç İçe) Bloklar
-------------------------------------------

declare
    v_disari varchar2(50) := 'Dıştaki Değişken';
begin

    declare
        v_iceri varchar2(50) := 'İçteki Değişken';
    begin
        dbms_output.put_line(v_disari);
        dbms_output.put_line(v_iceri);
    end;

    dbms_output.put_line(v_disari);
end;

-------------------------------------------
--Değişkenlerin Kapsam Alanı
-------------------------------------------

declare
    v_ulke varchar2(50)  := 'Dış - Türkiye';
    v_sehir varchar2(50) := 'Dış - Ankara';
begin

    declare
        v_sehir varchar2(50) := 'İç - İstanbul';
        v_ilce varchar2(50)  := 'İç - Çekmeköy';
    begin
        dbms_output.put_line(v_ulke);
        dbms_output.put_line(v_sehir);
        dbms_output.put_line(v_ilce);
    end;

    dbms_output.put_line(v_sehir);
--    dbms_output.put_line(v_ilce);

end;

-------------------------------------------

declare
    v_sayi1 number := 123;
    v_sayi2 number := 456;
begin

    declare
        v_carpim number;
    begin
        v_carpim := v_sayi1 * v_sayi2;
        dbms_output.put_line('Çarpım: '||v_carpim);
    end;

    declare
        v_toplam number;
    begin
        v_toplam := v_sayi1 + v_sayi2;
        dbms_output.put_line('Toplam: '||v_toplam);
    end;

end;
```

```
-----------------------------------------
--Nested Bloklarda Qualifier Kullanımı
-----------------------------------------

begin <<outer>>
declare
    v_ulke varchar2(50)  := 'Dış - Türkiye';
    v_sehir varchar2(50) := 'Dış - Ankara';
begin

    declare
        v_sehir varchar2(50) := 'İç - İstanbul';
        v_ilce varchar2(50)  := 'İç - Çekmeköy';
    begin
        dbms_output.put_line(v_ulke);
        dbms_output.put_line(v_sehir);
        dbms_output.put_line(outer.v_sehir);
        dbms_output.put_line(v_ilce);
    end;

end;
end outer;
```

```
-------------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------------

-----------------------------------------
--PL/SQL İçinde Kullanılan SQL Fonksiyonları
-----------------------------------------

declare
    v_customer_id   orders.customer_id%type;
    v_order_date    orders.order_date%type;
    v_shipped_date  orders.shipped_date%type;
    v_ship_via      varchar2(50);
    v_freight       orders.freight%type;
    v_output_text   varchar2(500);
    c_new_line      char(1) := CHR(10);

begin

    select
        customer_id, order_date, shipped_date,
        decode(ship_via, 1, 'Airway', 2, 'Seaway', 3, 'Roadway'), freight
        into v_customer_id, v_order_date, v_shipped_date, v_ship_via, v_freight
    from orders
    where order_id = 10538;

    v_output_text :=
        'Customer ID: '  || lower(v_customer_id) || c_new_line ||
        'Order Month: '  || to_char(v_order_date, 'Month') || c_new_line ||
        'Shiiped Date: ' || to_char(v_shipped_date, 'dd.mm.yyyy') || c_new_line ||
        'Ship Via: '     || v_ship_via || c_new_line ||
        'Order Amount: ' || to_char(round(v_freight));

    dbms_output.put_line(v_output_text);
end;

-----------------------------------------
--Değişkenlerin Kapsam Alanı
-----------------------------------------

declare
    v_net_price smallint;
    v_brand varchar2(20) := &brand;
begin

    declare
       v_price smallint;
       v_discount smallint;
    begin
        select price, discount into v_price, v_discount
        from cars
        where brand = v_brand;

        v_net_price := v_price - v_discount;
    end;

     dbms_output.put_line(v_brand ||'''s net price is: '|| v_net_price);
end;
/
```

```
-------------------------------------------
--SELECT İfadesinin Kullanımı
-------------------------------------------

declare
    v_ps_ismi varchar2(70);
begin

    select ad||' '||soyad
    into   v_ps_ismi
    from personel
    where personel_id = 5007;

    dbms_output.put_line('Personel İsmi: '||v_ps_ismi);

end;

-------------------------------------------

declare
    v_ps_ismi  varchar2(70);
    v_ps_unvan varchar2(30);
    v_ps_maas  number;
begin

    select ad||' '||soyad, unvan, maas
    into   v_ps_ismi, v_ps_unvan, v_ps_maas
    from personel
    where personel_id = 5007;

    dbms_output.put_line('Personel İsmi: '  ||v_ps_ismi);
    dbms_output.put_line('Personel Unvanı: '||v_ps_unvan);
    dbms_output.put_line('Personel Maaşı: ' ||v_ps_maas);

end;

-------------------------------------------
--Belirsizliklerin Kaldırılması
-------------------------------------------

declare
    ad    varchar2(70);
    unvan varchar2(30);
    maas  number;
    personel_id number;
begin

    select ad, unvan, maas
    into   ad, unvan, maas
    from personel
    where personel_id = 5007;

    delete from personel where personel_id = personel_id
end;
```

```
-------------------------------------------
--PL/SQL İçinde Verileri Değiştirme - INSERT
-------------------------------------------

declare
    v_max_id number;
begin

    select max(dept_id) into v_max_id
    from departman;

    insert into departman (dept_id, dept_ismi)
    values(v_max_id +1, 'Müşteri Elde Tutma');

    commit;
end;

-------------------------------------------
--PL/SQL İçinde Verileri Değiştirme - UPDATE
-------------------------------------------

declare
    v_artis_orani number := 21;

begin

    update personel
    set maas = maas + maas * (v_artis_orani / 100);

    .....

    --commit / rollback;

end;

-------------------------------------------
--PL/SQL İçinde Verileri Değiştirme - DELETE
-------------------------------------------

declare
    v_dept_id departman.dept_id%type := 100;

begin

    delete from departman
    where dept_id = v_dept_id;

end;

-------------------------------------------
--SQL Cursor (Implicit) - SQL%ROWCOUNT
-------------------------------------------

declare
    v_unvan personel.unvan%type := 'UZMAN';
    v_silinen_sayi number;

begin

    delete from personel
    where unvan = v_unvan;

    v_silinen_sayi := SQL%ROWCOUNT;

    dbms_output.put_line('Silinen Kayıt Sayısı: ' || v_silinen_sayi);

end;
```

```
-------------------------------------------
--SQL Cursor Özellikleri (Implicit) - SQL%FOUND
-------------------------------------------

declare
    v_sayi number;

begin

    update konum set konum_adi = 'İstanbul Çekmeköy'
    where konum_id = 5;

    v_sayi := SQL%ROWCOUNT;

    if sql%found then
       dbms_output.put_line('Güncellenen Kayıt Sayısı: ' || v_sayi);
    else
       dbms_output.put_line('Kayıt Bulunamadı!!');
    end if;

end;

-------------------------------------------
--SQL Cursor Özellikleri (Implicit) - Dikkat
-------------------------------------------

declare
    v_ps_ismi varchar2(70);
begin

    select ad||' '||soyad
    into   v_ps_ismi
    from personel
    where personel_id = 6000;

    if sql%notfound then
       dbms_output.put_line('Kayıt bulunamadı');
    else
       dbms_output.put_line('Kayıt bulundu');
    end if;

end;
```

```
-------------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------------

-----------------------------------------
--PL/SQL İçinde Verileri Değiştirme
-----------------------------------------

declare
    v_max_id smallint;
    v_new_id smallint;
    v_brand varchar2(10) := 'Opel';
    v_price smallint := 12000;
    v_discount smallint;

begin

    select max(id) into v_max_id from cars;

    insert into cars values(v_max_id + 1, v_brand, v_price, null)
    returning id into v_new_id;

    v_discount := v_price * 0.05;

    update cars set discount = v_discount where id = v_new_id;

    dbms_output.put_line('ID: '|| v_new_id || ', Brand: ' ||
        v_brand ||', Price: '  || v_price  || ', Discount: ' || v_discount);

    delete from cars where id = v_new_id;
end;
/

-----------------------------------------
--SQL Cursor Özellikleri
-----------------------------------------

begin

    update student set course_name = 'Accountancy and Finance'
    where course_name = 'Economics';

    if sql%notfound then
       dbms_output.put_line('Herhangi bir satır güncellenmedi, lütfen kontrol ediniz! ');
    else
       dbms_output.put_line('Güncellenen Kayıt Sayısı: '|| SQL%ROWCOUNT);
    end if;

end;
/
```

```
-------------------------------------------
--IF-ELSE İfadesinin Kullanımı
-------------------------------------------

declare
    maas number := 3500;
begin
    if maas < 5000 then
        dbms_output.put_line('Düşük maaş');
    end if;
end;

-------------------------------------------

declare
    maas number := 6000;
begin

    if maas < 5000 then
        dbms_output.put_line('Düşük Maaş');
    else
        dbms_output.put_line('Yüksek Maaş');
    end if;

end;

-------------------------------------------
--IF-ELSIF İfadesinin Kullanımı
-------------------------------------------

declare
    maas number := 6000;
begin

    if maas < 5000 then

        dbms_output.put_line('Düşük Maaş');

    elsif maas between 5000 and 10000 then

        dbms_output.put_line('Orta Maaş');

    else
        dbms_output.put_line('Yüksek Maaş');

    end if;

end;

-------------------------------------------
--IF-ELSE İfadesinde NULL
-------------------------------------------

declare
    maas number;

begin

    if maas < 5000 then
        dbms_output.put_line('Düşük Maaş');
    else
        dbms_output.put_line('Yüksek Maaş');
    end if;

end;
```

```
-------------------------------------------
--CASE İfadesi
-------------------------------------------

declare
    v_derece number := &derece;
    v_hava_durumu varchar2(50);

begin

    v_hava_durumu :=
        case
            when v_derece < 0               then 'Çok soğuk'
            when v_derece between 0 and 15  then 'Soğuk'
            when v_derece between 16 and 26 then 'Ilık'
            when v_derece > 26              then 'Sıcak'
        end;

    dbms_output.put_line('Sıcaklık: ' ||v_derece || ' derece,
      Hava Durumu: '||v_hava_durumu);

end;

-------------------------------------------

declare
    v_mevsim varchar2(20) := 'Sonbahar';
    v_hava_durumu varchar2(60);

begin

    v_hava_durumu :=
        case v_mevsim
            when 'İlkbahar' then 'Bitkiler yeniden canlanır, sıcaklık: 15-25'
            when 'Yaz'      then 'İşte tatil zamanı, sıcaklık: 25-35'
            when 'Sonbahar' then 'Doğadaki renk cümbüşü inanılmazdır, sıcaklık: 10-20'
            when 'Kış'      then 'Beyaz, soğuk ama çok güzeldir, sıcaklık: -25-10'
            else 'Bu bir mevsim değil!'
        end;

    dbms_output.put_line(v_mevsim||' - '||v_hava_durumu);

end;

-------------------------------------------

declare
    v_not   char(1);
    v_sonuc varchar2(20);
begin

    v_not := 'B';

    case v_not
        when 'A' then v_sonuc := 'Mükemmel' ;
        when 'B' then v_sonuc := 'Çok İyi' ;
        when 'C' then v_sonuc := 'İyi' ;
        when 'D' then v_sonuc := 'Yetersiz' ;
        when 'F' then v_sonuc := 'Zayıf' ;
    else
        v_sonuc := 'Böyle bir not yok' ;
    end case;

    dbms_output.put_line( v_sonuc );
end;

-------------------------------------------
```

```
declare
    v_satis_degeri  number;
    v_komisyon      number;
begin

    v_satis_degeri := 150000;
    case
        when v_satis_degeri > 200000 then
             v_komisyon := 0.2;
        when v_satis_degeri >= 100000 and v_satis_degeri < 200000 then
             v_komisyon := 0.15;
        when v_satis_degeri >= 50000  and v_satis_degeri < 100000 then
             v_komisyon := 0.1;
        when v_satis_degeri > 30000 then
             v_komisyon := 0.05;
        else
            v_komisyon := 0;
    end case;

    dbms_output.put_line( 'Komisyon değeri: %' || v_komisyon * 100);
end;

-------------------------------------------
--NULL Değeri İle Çalışma
-------------------------------------------

declare
    v_sayi1 number;
    v_sayi2 number;
    v_sayi3 number := 70;

begin

    v_sayi1 := 100;
    if v_sayi1 > v_sayi2 then
        dbms_output.put_line('Kontrol-1');
    end if;

    v_sayi2 := 80;
    if v_sayi1 > v_sayi2 then
        dbms_output.put_line('Kontrol-2');
    end if;

    if (v_sayi1 > v_sayi2) and (v_sayi1 > v_sayi3) then
        dbms_output.put_line('Kontrol-3');
    end if;
end;

-------------------------------------------
--Basic Döngüler
-------------------------------------------

set serveroutput on;
declare
    v_sayac    pls_integer := 0;
begin

    loop
        v_sayac := v_sayac + 1;
        dbms_output.put_line(v_sayac || '.sayı');

        exit when v_sayac = 10;

    end loop;

end;
```

```
-------------------------------------------

set serveroutput on;
declare
    v_sayac pls_integer := 0;
    v_ps_id pls_integer;
    v_ad    varchar2(50);
    v_soyad varchar2(40);

begin
    loop
        v_sayac := v_sayac + 1;

        v_ps_id := dbms_random.value(5000, 5020);

        select ad, soyad into v_ad, v_soyad
        from personel where personel_id = v_ps_id;

        dbms_output.put_line(v_ps_id || ': ' || v_ad || ' '|| v_soyad);

        exit when v_sayac = 10;
    end loop;
end;

-------------------------------------------
--While Döngüler
-------------------------------------------

set serveroutput on;
declare
    v_sayac    pls_integer := 0;

begin

    while v_sayac < 10 loop

        v_sayac := v_sayac + 1;
        dbms_output.put_line(v_sayac || '.sayı');

    end loop;

end;

-------------------------------------------

declare
    v_grupsayisi  number := 0;
    v_deger number := 10000;
    v_limit integer := 23;
    islemtamam boolean := false;
begin

    dbms_output.put_line('Değer:'|| v_deger);
    while islemtamam = false loop
        v_grupsayisi := v_grupsayisi + 1;

        if v_deger <= v_limit then
            islemtamam := true;
        else
            v_deger := v_deger - v_limit;
        end if;

    end loop;
    dbms_output.put_line('Limit:'|| v_limit);
    dbms_output.put_line('Grup Sayısı: '|| v_grupsayisi);
end;
```

```
-------------------------------------------
--FOR Döngüler
-------------------------------------------

begin

    for i in 1..10 loop

        dbms_output.put_line(i || '.sayı');

    end loop;

end;

-------------------------------------------

declare
    v_str varchar2(40) := 'PL/SQL-Eğitimi-Nasıl-Gidiyor?';
begin

    for i in reverse 1..length(v_str) loop

        dbms_output.put_line(substr(v_str, i, 1));

    end loop;

end;

-------------------------------------------

declare
    v_min_id departman.dept_id%type;
    v_dep_ismi departman.dept_ismi%type;

begin

    select min(dept_id) into v_min_id from departman;

    for i in 1..10 loop

        select dept_ismi into v_dep_ismi from departman
        where dept_id = v_min_id + i;

        dbms_output.put_line(v_min_id + i || ': ' || v_dep_ismi);

    end loop;

end;

-------------------------------------------
--Döngüleri Karşılaştırma
-------------------------------------------

    loop

        v_sayac := v_sayac + 1;
        dbms_output.put_line(v_sayac || '.sayı');

        exit when v_sayac = 10;

    end loop;
```

```
-------------------------------------------

    while v_sayac < 10 loop

        v_sayac := v_sayac + 1;
        dbms_output.put_line(v_sayac || '.sayı');

    end loop;

-------------------------------------------

begin

    for i in 1..10 loop

        dbms_output.put_line(i || '.sayı');

    end loop;

end;

-------------------------------------------
--İç İçe Döngüler
-------------------------------------------

declare
    i   pls_integer := 0;
    j   pls_integer := 0;
begin

    loop
        j := 1;
        i := i + 1;

        loop
            dbms_output.put_line(i || ' * ' || j || ' = '  || i*j);
            j := j + 1;
            exit when j = 4;
        end loop;

        exit when i = 4;
    end loop;
end;

-------------------------------------------

declare
   i number := 2;
   j number;

begin
    loop
        j:= 2;
        loop
            exit when ((mod(i, j) = 0) or (j = i));
            j := j +1;
        end loop;

        if (j = i) then
            dbms_output.put_line(i || ' : asal sayıdır');
        end if;

        i := i + 1;
        exit when i = 50;
    end loop;
end;
```

```
-------------------------------------------

begin
    <<dis_dongu>>
    for v_dis_sayac in 1..2 loop

        <<ic_dongu>>
        for v_ic_sayac in 1..4 loop

            dbms_output.put_line('Dış döngü: ' || v_dis_sayac || ', İç döngü: ' || v_ic_sayac);
            exit dis_dongu when v_ic_sayac = 3;

        end loop ic_dongu;

    end loop dis_dongu;
end;

-------------------------------------------
--Continue İfadesi
-------------------------------------------

declare
    a integer := 0;
begin

    while a < 20 loop

        a := a + 1;

        if mod(a,5) = 0 then
           continue;
        end if;

        dbms_output.put_line('a: '|| a);

    end loop;
end;

-------------------------------------------

begin

    for sayac in 1 .. 20 loop

        if mod( sayac, 2 ) = 1 then
            -- tek sayıları atla
            continue;
        end if;

        dbms_output.put_line(sayac);
    end loop;

end;
```

```
-------------------------------------------
--Continue When İfadesi
-------------------------------------------

begin

    for sayac in 1 .. 20 loop
        continue when mod( sayac, 2 ) = 0;
        dbms_output.put_line(sayac);
    end loop;

end;

-------------------------------------------
--Continue Label İfadesi
-------------------------------------------

declare
    v_toplam number := 0;
begin
    <<dis_loop>>
    for sayac1 in 1 .. 5 loop

        v_toplam := v_toplam + 1;
        dbms_output.put_line('Dış döngü: ' || v_toplam);

        for sayac2 in 1..5 loop

            v_toplam := v_toplam + 1;
            dbms_output.put_line('İç döngü: ' || v_toplam);

            continue dis_loop when mod(v_toplam, 2) = 0;

        end loop;

    end loop dis_loop;
end;
```

```
-------------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------------

-----------------------------------------
--CASE İfadesi
-----------------------------------------

declare
    v_id smallint := 2;
    v_tax smallint;
    v_price smallint;
    v_brand varchar2(50);

begin

    select brand, price into v_brand, v_price
    from cars where id = v_id;

    v_tax :=
        case
            when v_price between 5000 and 10000   then v_price * 0.1
            when v_price between 10001 and 100000 then v_price * 0.3
            when v_price > 100000                 then v_price * 0.6
        end;

    dbms_output.put_line('Brand: ' ||v_brand || ' , Price: '
        || v_price || ', Tax: : '||v_tax);

end;

-----------------------------------------
--Döngüler
-----------------------------------------

--ALıştırma-1

declare

    ilk_sayi number := 0;
    ikinci_sayi number := 1;
    toplam number;

    adet number := 10;
    i number;

begin

    for i in 2..adet
    loop
        toplam := ilk_sayi + ikinci_sayi;

        ilk_sayi := ikinci_sayi;
        ikinci_sayi := toplam;

        dbms_output.put_line(toplam);
    end loop;

end;
```

```
--ALıştırma-2
--BASIC-------------------------------------

declare
    v_tarih date;
    v_sayac pls_integer := 0;
begin

    v_tarih := to_date('02.01.2024', 'dd.mm.yyyy');

    loop

        if v_tarih != to_date('23.04.2024','dd.mm.yyyy') then

            v_sayac := v_sayac + 1;
            dbms_output.put_line(v_sayac || '. Toplantı: '||
                to_char(v_tarih, 'dd.mm.yyyy'));

        end if;

        v_tarih := v_tarih + 14;
        exit when v_sayac = 10;

    end loop;

end;

--WHILE-------------------------------------
declare
    v_tarih date;
    v_sayac pls_integer := 0;
begin

    v_tarih := to_date('02.01.2024', 'dd.mm.yyyy');

    while v_sayac < 10 loop

        if v_tarih != to_date('23.04.2024','dd.mm.yyyy') then

            v_sayac := v_sayac + 1;
            dbms_output.put_line(v_sayac || '. Toplantı: ' ||
                to_char(v_tarih, 'dd.mm.yyyy'));

        end if;

        v_tarih := v_tarih + 14;
        --exit when v_sayac = 10;

    end loop;

end;
```

```
--FOR-------------------------------------
declare
    v_tarih date;
    v_sayac pls_integer := 0;
begin

    v_tarih := to_date('02.01.2024', 'dd.mm.yyyy');

    for i in 1..11 loop

        if v_tarih != to_date('23.04.2024', 'dd.mm.yyyy') then

            v_sayac := v_sayac + 1;
            dbms_output.put_line(v_sayac || '. Toplantı: ' ||
                to_char(v_tarih, 'dd.mm.yyyy'));

        end if;

        v_tarih := v_tarih + 14;
        --exit when v_sayac = 10;

    end loop;

end;

-----------------------------------------
--Continue İfadesi
-----------------------------------------

declare
    v_date1 date := to_date('19.04.2024','dd.mm.yyyy');
    v_date2 date := to_date('20.05.2024','dd.mm.yyyy');
    v_control boolean := true;
    v_holiday_control smallint;
begin

    while v_control loop

        v_date1 := v_date1 + 1;

        if to_char(v_date1,'d') not in (6,7) then

            select count(*) into v_holiday_control
            from holidays
            where holiday_day = to_char(v_date1,'dd')
                and holiday_month = to_char(v_date1,'mm');

            if v_holiday_control = 0 then
                dbms_output.put_line('Çalışma günü: ' || v_date1);
            else
                dbms_output.put_line('Resmi tatil: ' || v_date1);
                continue;
            end if;
        else
            dbms_output.put_line('Hafta sonu: ' || v_date1);
            continue;
        end if;

        if v_date1 = v_date2 then
            v_control := false;
        end if;

    end loop;
end;
```

```
-------------------------------------------
--PL/SQL Records
-------------------------------------------

declare
    type type_perso is record
        (
            v_adi varchar2(30),
            v_soyadi personel.soyad%type,
            v_maas number,
            v_izin_gunu pls_integer
        );

    v_personel type_perso;

begin
    v_personel.v_adi := 'Ekrem';
    v_personel.v_soyadi := 'Tiryaki';
    v_personel.v_maas := 2000;
    v_personel.v_izin_gunu := 25;

    dbms_output.put_line('Adı: '||v_personel.v_adi);
    dbms_output.put_line('Soyadı: '||v_personel.v_soyadi);
    dbms_output.put_line('Maaşı: '||v_personel.v_maas);
    dbms_output.put_line('İzin Günü: '||v_personel.v_izin_gunu);
end;

-------------------------------------------

declare
    type type_perso is record
        (
            ad        personel.ad%type,
            soyad     personel.soyad%type,
            maas      personel.maas%type,
            giris     personel.giris_tarihi%type
        );

    v_personel type_perso;

begin

    select ad, soyad, maas, giris_tarihi into v_personel
    from personel where personel_id = 5010;

    dbms_output.put_line('Adı: '|| v_personel.ad);
    dbms_output.put_line('Soyadı: '|| v_personel.soyad);
    dbms_output.put_line('Maaşı: '|| v_personel.maas);
    dbms_output.put_line('İzin Günü: '|| v_personel.giris);
end;
```

```
-----------------------------------------------

declare
   type kitap_type is record
      (
        baslik varchar(50),
        yazar varchar(50),
        kitap_id number
      );
   kitap1 kitap_type;
   kitap2 kitap_type;
begin

   kitap1.baslik  := 'PL/SQL Programming';
   kitap1.yazar := 'Tuncay Tiryaki';
   kitap1.kitap_id := 123456;

   kitap2.baslik := 'SQL Tunning';
   kitap2.yazar := 'Sinem Bulut';
   kitap2.kitap_id := 659847;

   dbms_output.put_line('Kitap 1 baslik : '|| kitap1.baslik);
   dbms_output.put_line('Kitap 1 yazar : '|| kitap1.yazar);
   dbms_output.put_line('Kitap 1 kitap_id : ' || kitap1.kitap_id);

   dbms_output.put_line('Kitap 2 baslik : '|| kitap2.baslik);
   dbms_output.put_line('Kitap 2 yazar : '|| kitap2.yazar);
   dbms_output.put_line('Kitap 2 kitap_id : '|| kitap2.kitap_id);
end;

-----------------------------------------------
--PL/SQL Nested Records
-----------------------------------------------

declare
  type adres_type is record
  (
    ilce varchar2(255),
    sehir varchar2(100),
    ulke varchar2(100)
  );
  type musteri_type is record
  (
      musteri_ismi varchar2(100),
      teslim_adresi adres_type,
      fat_adresi adres_type
  );
  v_musteri musteri_type;
begin

  v_musteri.musteri_ismi := 'Faruk Keskin';
  v_musteri.teslim_adresi.ilce := 'Yenimahalle';
  v_musteri.teslim_adresi.sehir := 'Ankara';
  v_musteri.teslim_adresi.ulke := 'Türkiye';
  -- Fatura adresi ve teslim adresi aynı ise
  v_musteri.fat_adresi := v_musteri.teslim_adresi;
end;
```

```
-------------------------------------------
--%ROWTYPE Özelliği
-------------------------------------------

declare
    v_perso personel%rowtype;

begin
    select * into v_perso from personel
    where personel_id = 5060;

    dbms_output.put_line(v_perso.ad    || ' '  ||
                v_perso.soyad || ', ' ||
                v_perso.unvan);
end;

-------------------------------------------
--%ROWTYPE Özelliği - Select
-------------------------------------------

declare
    type kisi_type is record
    (
        ad_soyad      varchar2(100),
        toplam_gelir  number default 500,
        per_satir     personel%rowtype
    );

    v_calisan kisi_type;

begin
    select * into v_calisan.per_satir from personel
    where personel_id = 5020;

    v_calisan.ad_soyad := v_calisan.per_satir.ad || ' '
    || v_calisan.per_satir.soyad;

    v_calisan.toplam_gelir := v_calisan.per_satir.prim +
    v_calisan.per_satir.maas;

    dbms_output.put_line(v_calisan.ad_soyad || ', '
    || v_calisan.toplam_gelir);
end;

-------------------------------------------
--%ROWTYPE Özelliği - Insert
-------------------------------------------

create table personel_ayrilan as
select personel_id, ad, soyad, dept_id, cikis_tarihi
from personel where 1=0;

declare
    v_per_ayr personel_ayrilan%rowtype;
begin

    select personel_id, ad, soyad,
           dept_id, cikis_tarihi into v_per_ayr
    from personel where personel_id = 5020;

    insert into personel_ayrilan values v_per_ayr;
    commit;

end;
```

```
-------------------------------------------
--%ROWTYPE Özelliği - Update
-------------------------------------------

declare
    v_per_ayr personel_ayrilan%rowtype;
begin

    select * into v_per_ayr
    from personel_ayrilan where personel_id = 5020;

    v_per_ayr.cikis_tarihi := sysdate;

    update personel_ayrilan set ROW = v_per_ayr
    where personel_id = 5020;

    commit;

end;

-------------------------------------------
--Associative Arrays
-------------------------------------------

SET SERVEROUTPUT ON
DECLARE
  TYPE ulke IS TABLE OF VARCHAR2(50)
    INDEX BY VARCHAR2(5);

  t_ulkeler ulke;
BEGIN

  t_ulkeler('TR') := 'Türkiye';
  t_ulkeler('UK') := 'United Kingdom';
  t_ulkeler('FR') := 'France';
  t_ulkeler('DE') := 'Germany';

  DBMS_OUTPUT.PUT_LINE('ISO kod "TR" = ' || t_ulkeler('TR'));
  DBMS_OUTPUT.PUT_LINE('ISO kod "DE" = ' || t_ulkeler('DE'));

END;

-------------------------------------------

set serveroutput on;
declare

    type il_bilgi is table of varchar2(50)
        index by pls_integer;

    v_iller il_bilgi;
    c_ys varchar2(6) := chr(13)||chr(10); --Yeni satır

begin

    v_iller(6)  := 'ANKARA';
    v_iller(16) := 'BURSA';
    v_iller(46) := 'KAHRAMANMARAŞ';

    dbms_output.put_line(v_iller(6) ||c_ys||
                         v_iller(16)||c_ys||
                         v_iller(46));

end;
```

```
-----------------------------------------
--Associative Arrays - %ROWTYPE
-----------------------------------------

set serveroutput on
declare
    type dep_table is table of departman%rowtype
        index by varchar2(5);

    t_dept dep_table;
begin

    select * into t_dept(1) from departman
    where dept_id = 100;

    dbms_output.put_line('Dep ID : ' || t_dept(1).dept_id
        || ', Dep İsmi : ' ||
        t_dept(1).dept_ismi);

end;

-----------------------------------------

declare
    type dep_table is table of departman%rowtype
        index by varchar2(5);

    t_dept dep_table;
begin

    for i in 1..10 loop
        select * into t_dept(i) from departman
        where dept_id = 100+i;
    end loop;

    for i in 1..10 loop
        dbms_output.put_line('Dep ID : ' || t_dept(i).dept_id
            || ', Dep İsmi : ' ||
            t_dept(i).dept_ismi);
    end loop;
end;
```

```
-------------------------------------------
--Associative Arrays - Records
-------------------------------------------

declare
    type per_bilgi_type is record
    (
        id pls_integer,
        isim varchar2(30),
        maas number
    );

    type personeller_type is table of per_bilgi_type
        index by pls_integer;

    v_perss personeller_type;
begin

    v_perss(1).id := 10;
    v_perss(1).isim := 'Ayşe';
    v_perss(1).maas := 5000;

    v_perss(2).id := 11;
    v_perss(2).isim := 'Mehmet';
    v_perss(2).maas := 7000;

    dbms_output.put_line(v_perss(1).id ||'-'||v_perss(1).isim||'-'||v_perss(1).maas);
    dbms_output.put_line(v_perss(2).id ||'-'||v_perss(2).isim||'-'||v_perss(2).maas);
end;

-------------------------------------------
--Collection Metodları
-------------------------------------------

declare
  type ulke is table of varchar2(30)
    index by varchar2(2);

  t_ulkeler ulke;
begin

  t_ulkeler('TR') := 'Türkiye';
  t_ulkeler('UK') := 'United Kingdom';
  t_ulkeler('FR') := 'France';
  t_ulkeler('DE') := 'Germany';
  t_ulkeler('US') := 'Amerika';
  t_ulkeler('AU') := 'Avustralya';

  dbms_output.put_line(t_ulkeler.first);
  dbms_output.put_line(t_ulkeler.last);
  dbms_output.put_line(t_ulkeler.count);
  dbms_output.put_line(t_ulkeler.prior('TR'));
  dbms_output.put_line(t_ulkeler.next('UK'));

  t_ulkeler.delete('FR');
  dbms_output.put_line(t_ulkeler.count);

  if not t_ulkeler.exists('FR') then
    dbms_output.put_line('FR silinmiş');
  end if;

end;
```

```
-------------------------------------------
--Associative Arrays - LOOP
-------------------------------------------

declare
    type ulke is table of varchar2(30)
        index by varchar2(2);

    t_ulkeler ulke;
    v_index varchar2(2);

begin

    t_ulkeler('TR') := 'Türkiye';
    t_ulkeler('UK') := 'United Kingdom';
    t_ulkeler('FR') := 'France';
    t_ulkeler('DE') := 'Germany';
    t_ulkeler('US') := 'Amerika';
    t_ulkeler('AU') := 'Avustralya';

    v_index := t_ulkeler.first;

    while v_index is not null loop

        dbms_output.put_line(v_index || ': '|| t_ulkeler(v_index));
        v_index := t_ulkeler.next(v_index);

    end loop;

end;

-------------------------------------------
--Nested Tables
-------------------------------------------

declare
    type isim_type is table of varchar2(30);

    v_isimler isim_type;
begin

    v_isimler := isim_type('Habibe', 'Ayşen', 'Esra');

    v_isimler.extend;
    v_isimler(v_isimler.count) := 'Yasemin';

    for i in v_isimler.first..v_isimler.last loop

        dbms_output.put_line(v_isimler(i));

    end loop;

end;
```

```
-------------------------------------------

declare
   type isim_type is table of varchar2(10);
   type derece_type is table of integer;

   isimler isim_type;
   dereceler derece_type;
   toplam integer;
begin

   isimler := isim_type('Sinem', 'Kadir', 'Kemal', 'Fatma', 'Şener');
   dereceler:= derece_type(98, 97, 78, 87, 92);
   toplam := isimler.count;

   dbms_output.put_line('Toplam öğrenci sayısı: '|| toplam);

   for i in 1 .. toplam loop

      dbms_output.put_line('Öğrenci: '||isimler(i)||', Derece: ' || dereceler(i));

   end loop;

end;

-------------------------------------------
--Varrays
-------------------------------------------

set serveroutput on;
declare
    type t_name_type is varray(2)
        of varchar2(20) not null;

    t_names t_name_type  := t_name_type('Sabri','Sinem');
    t_snames t_name_type := t_name_type();

Begin

    dbms_output.put_line('İsimler Sayısı: ' || t_names.count);
    dbms_output.put_line('Soyisimler Sayısı: ' || t_snames.count);

    dbms_output.put_line(t_names(1));
    dbms_output.put_line(t_names(2));

end;
```

```
-------------------------------------------

set serveroutput on;
declare
    type t_name_type is varray(2)
        of varchar2(20) not null;
    t_names t_name_type  := t_name_type('Sabri','Sinem');
    t_snames t_name_type := t_name_type();
begin
    dbms_output.put_line('İsimler Sayısı: ' || t_names.count);
    dbms_output.put_line('Soyisimler Sayısı: ' || t_snames.count);

    dbms_output.put_line(t_names(1));
    dbms_output.put_line(t_names(2));

    --t_snames(1) := 'Kurt'; Hata verir
    t_snames.extend; --Bir element ekliyoruz
    t_snames(1) := 'Kurt';

    t_snames.extend;
    t_snames(2) := 'Kedi';

    dbms_output.put_line(t_snames(1));
    dbms_output.put_line(t_snames(2));

end;

-------------------------------------------

declare
    type ay_type is varray(12)
        of varchar2(20) not null;

    type gun_type is varray(7)
        of varchar2(20) not null;

    v_aylar ay_type  := ay_type('Ocak','Şubat', 'Mart', 'Nisan', 'Mayıs',
        'Haziran', 'Temmuz', 'Ağustos', 'Eylül', 'Ekim', 'Kasım', 'Aralık');

    v_gunler gun_type := gun_type('Pazartesi', 'Salı', 'Çarşamba',
        'Perşembe', 'Cuma', 'Cumartesi', 'Pazar');

    v_ay_no simple_integer := 5;
    v_gun_no simple_integer := 2;

Begin

    dbms_output.put_line(v_aylar(v_ay_no));
    dbms_output.put_line(v_gunler(v_gun_no));

end;
```

```
------------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
------------------------------------------------------

-----------------------------------------
--PL/SQL Records
-----------------------------------------

declare
    type type_order_info is record
        (
            first_name    employees.first_name%type,
            last_name     employees.last_name%type,
            order_count   number,
            order_sum     number
        );

    v_order_info type_order_info;

begin

    select e.first_name, e.last_name,
        count(*) order_count,
        sum(freight) order_sum
        into v_order_info
    from employees e, orders o
    where e.employee_id = 3
        and o.employee_id = e.employee_id
    group by e.first_name, e.last_name;

    dbms_output.put_line('First name: '|| v_order_info.first_name);
    dbms_output.put_line('Last name: '|| v_order_info.last_name);
    dbms_output.put_line('Order count: '|| v_order_info.order_count);
    dbms_output.put_line('Order sum: '|| v_order_info.order_sum);
end;

-----------------------------------------
--%ROWTYPE Özelliği
-----------------------------------------

declare
    rt_product products%rowtype;
    rt_category categories%rowtype;
    rt_supplier suppliers%rowtype;

begin

    select * into rt_product from products
    where product_id = 1;

    select * into rt_category from categories
    where category_id = rt_product.category_id;

    select * into rt_supplier from suppliers
    where supplier_id = rt_product.supplier_id;

    dbms_output.put_line('Product Name: ' || rt_product.product_name);
    dbms_output.put_line('Product Unit Price: ' || rt_product.unit_price);
    dbms_output.put_line('Category Name: ' || rt_category.category_name);
    dbms_output.put_line('Supplier Name: ' || rt_supplier.company_name);
    dbms_output.put_line('Supplier City: ' || rt_supplier.city);

end;
```

```
-----------------------------------------
--Associative Arrays
-----------------------------------------

declare
    type cars_type is record
    (
        brand varchar2(30),
        price number,
        discount number
    );

    type cars_table_type is table of cars_type
        index by pls_integer;

    v_cars_info cars_table_type;
    v_cars_count pls_integer;
begin

    select count(*) into v_cars_count from cars;

    for i in 1..v_cars_count loop
        select brand, price, discount
        into v_cars_info(i) from cars
        where id = i;
    end loop;

    for i in 1..v_cars_count loop
        dbms_output.put_line('Brand : ' || v_cars_info(i).brand
            || ', Price : ' || v_cars_info(i).price
            || ', Discount : ' || nvl(v_cars_info(i).discount,0));
    end loop;
end;
```

```
-----------------------------------------
--Associative Arrays Loop
-----------------------------------------

declare
    type student_type is table of varchar2(50)
        index by varchar2(50);

    v_students student_type;
    v_students_count pls_integer;
    v_engineers_count pls_integer := 0;

    v_key varchar2(50);

    v_name varchar2(50);
    v_course_name varchar2(50);

begin

    select count(*)
    into v_students_count
    from student;

    for i in 1..v_students_count loop

        select name, course_name
        into v_name, v_course_name from student
        where id = i;

        v_students(v_name) := v_course_name;

    end loop;

    v_key := v_students.first;

    while v_key is not null loop

        if (lower(v_students(v_key)) like '%engineer%') then
            dbms_output.put_line(v_key || ': '|| v_students(v_key));
            v_engineers_count := v_engineers_count + 1;
        end if;

        v_key := v_students.next(v_key);

    end loop;

    dbms_output.put_line('-----------------------------------');
    dbms_output.put_line('All students count: ' || v_students.count);
    dbms_output.put_line('Engineer students count: ' || v_engineers_count);

end;
```

```
-----------------------------------------
--Nested Tables
-----------------------------------------

declare
   type ay_isim_type is table of varchar2(10);
   type ay_gun_type is table of integer;

   ay_isimleri ay_isim_type;
   ay_gunleri ay_gun_type;
begin

    ay_isimleri := ay_isim_type('Ocak', 'Şubat', 'Mart', 'Nisan', 'Mayıs',
        'Haziran', 'Temmuz', 'Ağustos', 'Eylül', 'Ekim');

    ay_gunleri := ay_gun_type(31, 28, 31, 30, 31, 30, 31, 31, 30, 31);

    ay_isimleri.extend(2);
    ay_isimleri(ay_isimleri.count-1) := 'Kasım';
    ay_isimleri(ay_isimleri.count) := 'Aralık';

    ay_gunleri.extend;
    ay_gunleri(ay_gunleri.count) := 30;

    ay_gunleri.extend;
    ay_gunleri(ay_gunleri.last) := 31;

    for i in ay_isimleri.first..ay_isimleri.last loop

        dbms_output.put_line(ay_isimleri(i)||' ayı gün sayısı: ' || ay_gunleri(i));

    end loop;

end;
```

```
-----------------------------------------
--Varrays
-----------------------------------------

declare
    type customer_info_type is record
    (
        customer_id customers.customer_id%type,
        company_name customers.company_name%type,
        city customers.city%type
    );
    v_customer_info customer_info_type;

    type customers_type is varray(5)
        of customer_info_type;

    v_customers customers_type := customers_type();

    type t_customerid_type is varray(3) of varchar2(20);
    v_customer_ids t_customerid_type  := t_customerid_type('BERGS','CACTU','TOMSP');

begin

    for i in v_customer_ids.first..v_customer_ids.last loop

        select customer_id, company_name, city into v_customer_info
        from customers
        where customer_id = v_customer_ids(i);

        v_customers.extend;
        v_customers(v_customers.last).customer_id := v_customer_info.customer_id;
        v_customers(v_customers.last).company_name := v_customer_info.company_name;
        v_customers(v_customers.last).city := v_customer_info.city;

    end loop;

    -- show all customers
    for j in v_customers.first..v_customers.last loop
        dbms_output.put_line(
            'Customer id: ' || v_customers(j).customer_id ||
            ', Comapny name: ' || v_customers(j).company_name ||
            ', City: ' || v_customers(j).city
        );
    end loop;

end;
```

```
-------------------------------------------
--Implicit Cursors
-------------------------------------------

set serveroutput on;
begin

    update personel
    set maas = 3000
    where personel_id = 5010;

    if sql%notfound then

        dbms_output.put_line('Personel bulunamadı');

    elsif sql%found then

        dbms_output.put_line('Güncellenen kayıt sayısı: ' || sql%rowcount);

    end if;

end;

-------------------------------------------
--Cursor'dan Data Getirme – Tek Satır
-------------------------------------------

declare
    cursor c_personel is
        select ad, maas from personel
        where personel_id = 5020;

    v_ad personel.ad%type;
    v_maas personel.maas%type;

begin

    open c_personel;
    fetch c_personel into v_ad, v_maas;

    dbms_output.put_line(v_ad|| ': '||v_maas);

end;

-------------------------------------------
--Cursor'dan Data Getirme – Çok Satır
-------------------------------------------

declare
    cursor c_personel is
        select ad, maas from personel
        where unvan = 'UZMAN';

    v_ad personel.ad%type;
    v_maas personel.maas%type;

begin

    open c_personel;

    loop
        fetch c_personel into v_ad, v_maas;
        exit when c_personel%notfound;
        dbms_output.put_line(v_ad|| ': '||v_maas);
    end loop;

end;
```

```
-------------------------------------------
--Cursor - Record Birlikte Kullanımı
-------------------------------------------

declare
    cursor c_personel is
        select ad, soyad, maas from personel
        where unvan = 'UZMAN';

    v_per_record c_personel%rowtype;

begin

    open c_personel;

    loop
        fetch c_personel into v_per_record;
        exit when c_personel%notfound;

        dbms_output.put_line(v_per_record.ad ||' '||
                             v_per_record.soyad || ': '||
                             v_per_record.maas);
    end loop;

    close c_personel;
end;

-------------------------------------------
--For Loop ile Cursor Kullanımı
-------------------------------------------

declare
    cursor c_personel is
        select ad, soyad, maas from personel
        where unvan = 'UZMAN';

begin

    for v_per_record in c_personel loop

        dbms_output.put_line(v_per_record.ad ||' '||
                             v_per_record.soyad || ': '||
                             v_per_record.maas);
    end loop;

end;

-------------------------------------------
--For Loop ile Cursor Kullanımı - Alt sorgular
-------------------------------------------

begin
    for v_row in (select * from konum) loop

        dbms_output.put_line(v_row.konum_id ||': '||
                             v_row.konum_adi);

    end loop;
end;
```

```
-------------------------------------------

begin
    for v_row in
        (select personel_id, ad, soyad, maas
         from personel where unvan = 'UZMAN')
    loop

        dbms_output.put_line(v_row.personel_id ||': '||
                             v_row.ad ||' '|| v_row.soyad ||' '||
                             v_row.maas);
    end loop;
end;

-------------------------------------------
--Cursor Özellikleri
-------------------------------------------

declare
    cursor c_personel is
        select ad, soyad, maas from personel
        where unvan = 'UZMAN';

    v_per_record c_personel%rowtype;

begin

    if not c_personel%isopen then
        open c_personel;
    end if;

    loop
        fetch c_personel into v_per_record;
        exit when c_personel%notfound or c_personel%rowcount > 12;

        dbms_output.put_line(c_personel%rowcount||': '||
                             v_per_record.ad ||' '||
                             v_per_record.soyad);

    end loop;
end;

-------------------------------------------
--Cursor'e Parametre Verme
-------------------------------------------

declare
    cursor c_personel (cv_unvan varchar2) is
        select ad, soyad, maas from personel
        where unvan = cv_unvan;
begin

    dbms_output.put_line('.....UZMANLAR.....');

    for v_per_record in c_personel('UZMAN') loop

        dbms_output.put_line(v_per_record.ad ||' '||
                             v_per_record.soyad);
        if c_personel%rowcount > 5 then exit; end if;

    end loop;

    dbms_output.put_line('.....MÜDÜRLER.....');

    for v_per_record in c_personel('MÜDÜR') loop

        dbms_output.put_line(v_per_record.ad ||' '||
```

```
                                v_per_record.soyad);
        if c_personel%rowcount > 5 then exit; end if;
    end loop;
end;

-------------------------------------------

declare
    cursor c_personel (cv_maas1 number, cv_maas2 number) is
        select ad, maas from personel
        where maas between cv_maas1 and cv_maas2;

begin

    dbms_output.put_line('.....Düşük Maaşlar....');

    for v_per_record in c_personel(1000, 3000) loop

        dbms_output.put_line(v_per_record.ad ||': '||
                                v_per_record.maas);
        if c_personel%rowcount > 3 then exit; end if;

    end loop;

    dbms_output.put_line('.....Yüksek Maaşlar.....');

    for v_per_record in c_personel(3001, 10000) loop

        dbms_output.put_line(v_per_record.ad ||': '||
                                v_per_record.maas);
        if c_personel%rowcount > 3 then exit; end if;
    end loop;
end;

-------------------------------------------
--Where Current Of İfadesi
-------------------------------------------

declare

    cursor crs_personel(v_unvan varchar2) is
        select personel_id, ad, soyad, maas
        from personel
        where unvan = v_unvan
        for update;

begin
    for row_prs in crs_personel('UZMAN') loop

        update personel set maas = maas*1.25
        where current of crs_personel;

    end loop;
end;
```

```
-------------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------------

-----------------------------------------
--Cursor'dan Data Getirme
-----------------------------------------

declare
    cursor c_emp is
        select
            title_of_courtesy || ' ' ||
            first_name        || ' ' ||
            last_name         || ' ' ||
            birth_date as emp_info
        from employees e;

    v_emp_info varchar(500);

begin

    open c_emp;
    loop
        fetch c_emp into v_emp_info;
        exit when c_emp%notfound;
        dbms_output.put_line(v_emp_info);
    end loop;

end;

-----------------------------------------
--Cursor - Record Birlikte Kullanımı
-----------------------------------------

DECLARE
    CURSOR c_cars IS
    SELECT
        brand, price, nvl(discount, 0) AS discount,
        price - nvl(discount, 0) AS net_price
    FROM cars;

    v_car_record   c_cars%rowtype;
    v_sum_price    NUMBER := 0;
    v_sum_discount NUMBER := 0;
BEGIN
    OPEN c_cars;
    LOOP
        FETCH c_cars INTO v_car_record;
        EXIT WHEN c_cars%notfound;

        v_sum_price := v_sum_price + v_car_record.price;
        v_sum_discount := v_sum_discount + v_car_record.discount;

        dbms_output.put_line(v_car_record.brand
                             || ', Price: '
                             || v_car_record.price
                             || ', Discount: '
                             || v_car_record.discount
                             || ', Net Price: '
                             || v_car_record.net_price);

    END LOOP;
    CLOSE c_cars;
    dbms_output.put_line('Sum price is: ' || v_sum_price);
    dbms_output.put_line('Sum discount is: ' || v_sum_discount);

END;
```

```
-----------------------------------------
--For Loop ile Cursor Kullanımı
-----------------------------------------
--Yöntem-1

declare
    cursor c_product is
        select segment as segment_name,
            count(*) product_count,
            sum(price) sum_price
        from product p, product_segment ps
        where p.segment_id = ps.id
        group by segment;

begin

    for v_product in c_product loop

        dbms_output.put_line(
            v_product.segment_name ||': Count: '||
            v_product.product_count || ', Sum Price: '||
            v_product.sum_price);
    end loop;

end;

--Yöntem-2

begin

    for v_product in
        (   select segment as segment_name,
                count(*) product_count,
                sum(price) sum_price
            from product p, product_segment ps
            where p.segment_id = ps.id
            group by segment
        )
    loop

        dbms_output.put_line(
            v_product.segment_name ||': Count: '||
            v_product.product_count || ', Sum Price: '||
            v_product.sum_price);
    end loop;

end;
```

```
-------------------------------------------
--Cursor'e Parametre Verme
-------------------------------------------

declare
    cursor c_student (cv_course_name varchar2) is
        select name, gender from student
        where course_name = cv_course_name;
begin

    dbms_output.put_line('------Computer Engineer------');

    for v_stud_record in c_student('Computer Engineer') loop

        dbms_output.put_line(v_stud_record.name ||', Gender: '||
                             v_stud_record.gender);
        if c_student%rowcount > 2 then exit; end if;

    end loop;

    dbms_output.put_line('');
    dbms_output.put_line('------Computer Science-----');

    for v_stud_record in c_student('Computer Science') loop

        dbms_output.put_line(v_stud_record.name ||', Gender: '||
                             v_stud_record.gender);
        if c_student%rowcount > 2 then exit; end if;

    end loop;

end;
```

```
-----------------------------------------
--Exception Örnek
-----------------------------------------

declare
    v_name varchar2(25);
begin
    select ad into v_name from personel
    where unvan = 'UZMAN';

    dbms_output.put_line(v_name);
end;

-----------------------------------------
--Exception Handling (Hata Yönetme)
-----------------------------------------

set serveroutput on;
declare
    v_name varchar2(25);
begin
    select ad into v_name from personel
    where unvan = 'UZMAN';

    dbms_output.put_line(v_name);

    exception
        when too_many_rows then
            dbms_output.put_line('Birden fazla satır geldi');
end;

-----------------------------------------
--Hata Yakalama Predefined
-----------------------------------------

declare
    v_name varchar2(25);
begin
    select ad into v_name from personel
--    where unvan = 'UZMAN';
    where personel_id = 5010;

    dbms_output.put_line(v_name);

    dbms_output.put_line('Bölme İşlemi: '|| 1/0);

    exception
        when TOO_MANY_ROWS then
            dbms_output.put_line('Birden fazla satır geldi');
        when OTHERS then
            dbms_output.put_line('Başka bir hata oluştu');
end;
```

```
-------------------------------------------
--Hata Yakalama Internally
-------------------------------------------

set serveroutput on;
declare

    e_null_insert exception;

    pragma exception_init(e_null_insert, -01400);

begin

    insert into bolgeler (bolge_kodu, bolge_adi)
    values(10, null);

    exception

        when e_null_insert then

            dbms_output.put_line('Null olmayan bir değer girmelisiniz!');

end;

-------------------------------------------
--Hata Yakalama - Fonksiyonlar
-------------------------------------------

create table hata_bilgileri
(
    tarih date,
    hata_kodu number,
    hata_bilgi varchar2(1000)
)

declare
    v_name varchar2(25);
    v_ecode number;
    v_emesg varchar2(1000);

begin
    select ad into v_name from personel
    where unvan = 'UZMAN';
--    where personel_id = 5010;

    dbms_output.put_line('Bölme İşlemi: '|| 1/0);

    exception
        when OTHERS then
            --rollback;
            v_ecode := SQLCODE;
            v_emesg := SQLERRM;
            insert into hata_bilgileri
            values (sysdate, v_ecode, v_emesg);
end;
```

```
-----------------------------------------
--Hata Yakalama - Kullanıcı Tanımlı
-----------------------------------------

declare
    v_maas number := &maas_degeri;
    e_maas_hatasi exception;

begin

    if v_maas > 20000 then
        raise e_maas_hatasi;
    end if;

    insert into personel (personel_id, ad, soyad, maas)
    values(1000, 'Esra', 'Yılmaz', v_maas);

    exception
        when e_maas_hatasi then
            dbms_output.put_line('Maaş değeri 20000 den büyük olamaz');
end;

-----------------------------------------

set serveroutput on;
declare
    v_dep pls_integer := 105;
    v_per_sayisi pls_integer;
    e_departman exception;

begin

    select count(*) into v_per_sayisi from personel
    where dept_id = v_dep;

    if v_per_sayisi < 5 then
        raise e_departman;
    end if;

    exception
        when e_departman then
            dbms_output.put_line('Dep. No: '|| v_dep ||', Personel sayısı: ' || v_per_sayisi);
end;
```

```
-------------------------------------------
--Alt Bloklarla Hata Yönetme
-------------------------------------------

declare
    e_departman exception;
    e_adet exception;

    cursor c_depart is
        select dept_id, count(*) adet
        from personel group by dept_id;
begin

    for row_d in c_depart loop
        begin

            if row_d.dept_id = 104 then
                raise e_departman;
            end if;

            dbms_output.put_line('[---]Dep. No: '|| row_d.dept_id ||',Adet: '||row_d.adet);

            if row_d.adet < 5 then
                raise e_adet;
            end if;

            exception
                when e_adet then
                dbms_output.put_line('[XXX]Dep. No: '|| row_d.dept_id ||',Adet: '||row_d.adet);
        end;
    end loop;

    exception
        when e_departman then
            dbms_output.put_line('!!! GÜVENLİK İHLALİ !!!');
end;

-------------------------------------------
--Raise_Application_Error
-------------------------------------------

declare
    v_deger number := &deger;
begin
    if v_deger >= 100 then
        raise_application_error(-20001, '100 den küçük bir değer girmelisiniz...');
    else
        dbms_output.put_line('Teşekkürler...');

    end if;
end;

-------------------------------------------

declare
    v_prim number;
begin

    select prim into v_prim from personel
    where personel_id = 5009;

    if v_prim is null then
        raise_application_error(-20005, 'Personel primi hak ediyor :)');
    else
        dbms_output.put_line('Prim Değeri:' || v_prim);

    end if;
end;
```

```
-----------------------------------------

declare
    e1 exception;
    pragma exception_init (e1, -20001);
    e2 exception;
    pragma exception_init (e2, -20002);
    e3 exception;
    pragma exception_init (e3, -20003);
    v_deger number := 1;

begin
    begin
        if v_deger = 1 then
            raise_application_error(-20001,'Hata Oluştu: No==> 1');
        elsif v_deger = 2 then
            raise_application_error(-20002,'Hata Oluştu: No==> 2');
        else
            raise_application_error(-20003,'Hata Oluştu: No==> 1 ve 2 değil');
        end if;
        -- iç blokta hata yakalanıyor
        exception
            when e1 then
                dbms_output.put_line('Hata Yakalandı: No==> 1');
    end;

    -- dış blokta hata yakalanıyor
    exception
        when e2 then
            dbms_output.put_line('Hata Yakalandı: No==> 2');
end;
```

```
-------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------

-----------------------------------------
--Hata Yakalama Predefined
-----------------------------------------

declare
    v_car_name varchar2(50) := 'Ferrari';
    v_mesaj varchar2(100);
begin


    select brand || ' is found in the table' into v_mesaj
    from cars where brand = v_car_name;

    dbms_output.put_line(v_mesaj);

    exception
        when NO_DATA_FOUND then
            dbms_output.put_line(v_car_name || ' is not found');

end;
/

-----------------------------------------
--Hata Yakalama - Kullanıcı Tanımlı
-----------------------------------------

declare
    v_budget number := 15000;
    v_car_brand varchar2(50) := 'Peugeot';
    v_car_price number;
    e_price_exceed exception;

begin

    select price into v_car_price from cars
    where brand = v_car_brand;

    if v_car_price > v_budget then
        raise e_price_exceed;
    else
        dbms_output.put_line('Tebirkler... Bu arabayı alabilirsiniz :)');
    end if;

    --Kredi işlemleri...

    exception
        when e_price_exceed then
            dbms_output.put_line('Üzgünüz! Bu arabanın fiyatı bütçenizi aşıyor :(');
end;
```

```
-----------------------------------------
--Alt Bloklarla Hata Yönetme - Alıştırma
-----------------------------------------

declare

    type v_colors_type is table of varchar2(50);
    v_colors v_colors_type;

    v_color varchar2(50);
begin

    v_colors := v_colors_type('Blue', 'Dark Green', 'Yellow');

    for i in v_colors.first..v_colors.last
    loop

        begin
            select color || ' is found in the table' into v_color
            from colors where color = v_colors(i);

            dbms_output.put_line(v_color);

            exception
                when NO_DATA_FOUND then
                    dbms_output.put_line(v_colors(i) || ' is not found');

        end;
    end loop;

end;
/
```

```
-----------------------------------------
--Raise_Application_Error
-----------------------------------------

declare
    e_stock_price_exceed_level1 exception;
    pragma exception_init (e_stock_price_exceed_level1, -20001);

    e_stock_price_exceed_level2 exception;
    pragma exception_init (e_stock_price_exceed_level2, -20002);

    e_stock_price_exceed_level3 exception;
    pragma exception_init (e_stock_price_exceed_level3, -20003);

    cursor c_products is
        select product_name,
            unit_price*units_in_stock as stock_price
        from products;
    v_prod_info varchar2(150);

begin

    for v_product_row in c_products loop

        v_prod_info := v_product_row.product_name||': '||v_product_row.stock_price;

        begin
            if v_product_row.stock_price between 2000 and 3000 then
                --raise e_stock_price_exceed_level1;
                raise_application_error(-20001,'Alarm!!! Stock Price Level1: ' || v_prod_info);
            elsif v_product_row.stock_price between 3000 and 4000 then
                --raise e_stock_price_exceed_level2;
                raise_application_error(-20002,'Alarm!!! Stock Price Level2: ' || v_prod_info);
            elsif v_product_row.stock_price > 4000 then
                raise_application_error(-20003,'Critical Alarm!!! Stock Price Level3: ' ||
v_prod_info);
            end if;

            exception
                when e_stock_price_exceed_level1 then
                    dbms_output.put_line('!!!Alarm!!! Stock Price Level1: ' || v_prod_info);
                when e_stock_price_exceed_level2 then
                    dbms_output.put_line('!!!Alarm!!! Stock Price Level2: ' || v_prod_info);
                when e_stock_price_exceed_level3 then
                    dbms_output.put_line(SQLERRM);
        end;
    end loop;

end;
```

```
-----------------------------------------
--Prosedürler - Örnek
-----------------------------------------

create procedure karesini_al
is
begin

    for i in reverse 1..10 loop

        dbms_output.put_line(lpad(i,2,' ') || ' ==> karesi : ' || i**2);

    end loop;

end;

-----------------------------------------

create or replace procedure uzman_yazdir is

    cursor c_personel is
        select ad, soyad, maas from personel
        where unvan = 'UZMAN';

begin

    for row_per in c_personel loop

        dbms_output.put_line(row_per.ad ||' '||
                             row_per.soyad || ': '||
                             row_per.maas);
    end loop;

end;

-----------------------------------------
--Prosedürler - Parametreli
-----------------------------------------

create or replace procedure personel_yazdir (p_unvan varchar2)
is

    cursor c_personel is
        select ad, soyad, maas from personel
        where unvan = p_unvan;

begin

    for row_per in c_personel loop

        dbms_output.put_line(row_per.ad    || ' ' ||
                             row_per.soyad || ': '||
                             row_per.maas  || ', '||
                             row_per.unvan);
    end loop;

end;
```

```
-----------------------------------------

create or replace procedure konum_ekle
    (
        p_konum_id konum.konum_id%type,
        p_konum_adi varchar2,
        p_il_kodu number
    )
is

begin

    insert into konum
    values(p_konum_id, p_konum_adi, p_il_kodu);

    commit;

end;

-----------------------------------------

create or replace procedure personel_bilgi
    (
        p_personel_id number
    )
is
    v_ad varchar2(50);
    v_unvan varchar2(30);
    v_maas number;
begin

    select ad, unvan, maas into v_ad, v_unvan, v_maas
    from personel
    where personel_id = p_personel_id;

    dbms_output.put_line(v_ad || ', ' || v_unvan || ': ' || v_maas);

end;

-----------------------------------------
--Prosedürler – Parametreli (OUT)
-----------------------------------------

create or replace procedure personel_bilgi
    (
        p_personel_id   in  number,
        p_ad            out varchar2,
        p_maas          out number
    )
is
begin
    select ad, maas into p_ad, p_maas
    from personel
    where personel_id = p_personel_id;
end;

declare
    v_ad varchar2(50);
    v_maas number;

begin
    personel_bilgi(5020, v_ad, v_maas);
    dbms_output.put_line(v_ad || ': ' || v_maas);
end;
```

```
-------------------------------------------
--Prosedürler - Parametreli (IN OUT)
-------------------------------------------

create or replace procedure telno_formatla
    (
      p_telno  IN OUT  varchar2
    )
is
begin

    p_telno := '(' ||
               substr(p_telno,1,3) || ') '||
               substr(p_telno,4,3) || ' ' ||
               substr(p_telno,7,2) || ' ' ||
               substr(p_telno,9,2);

end;

declare
    v_telefon_no varchar2(20) := '5859638541';
begin

    telno_formatla(v_telefon_no);
    dbms_output.put_line(v_telefon_no);

end;

-------------------------------------------
--Prosedürler - Parametreleri Dinamik Verme
-------------------------------------------

create table faaliyetler
(
     faaliyet_id   number,
     faaliyet      varchar2(100),
     faaliyet_gunu date
);

create procedure faaliyet_ekle
    (
        p_id  number := -1,
        p_adi varchar2 default 'Doğum günü',
        p_gunu date default sysdate
    )
is
begin

    insert into faaliyetler values(p_f_id, p_f_adi, p_f_gunu);
    commit;

end;

exec faaliyet_ekle;
exec faaliyet_ekle(7);
exec faaliyet_ekle(10, 'Haftalık raporlar');
exec faaliyet_ekle(1, 'Yılbaşı partisi', to_date('31.12.2020', 'dd.mm.yyyy'));
exec faaliyet_ekle(p_id => 2, p_adi => 'Dünya yazılımcılar günü', p_gunu => sysdate+10);
exec faaliyet_ekle(p_gunu => sysdate-20, p_id => 3, p_adi => 'Ramazan bayramı');
exec faaliyet_ekle(4, 'Eşimin doğum günü partisi', p_gunu => add_months(sysdate, 2));
exec faaliyet_ekle(p_id=>5, 'Hoşgeldin bahar pikniği', p_gunu => sysdate-22); --!!
exec faaliyet_ekle(p_id=>6, p_adi => 'Proje kapanış etkinliği', sysdate);  --!!
exec faaliyet_ekle(p_adi ='Günlük faaliyetler');
exec faaliyet_ekle(p_gunu => to_date('01.01.2021','dd.mm.yyyy'));
```

```
-----------------------------------------
--Fonksiyonlar - Örnek
-----------------------------------------

create or replace function f_faktoryel (p_sayi number)
return number
is
    v_sonuc number := 1;

begin

    for i in reverse 1..p_sayi loop

        v_sonuc := v_sonuc * i;

    end loop;

    return v_sonuc;
end;

-----------------------------------------

create or replace function f_ucret_duzeyi(p_id number)
return varchar2
is
    v_ucret_duzey varchar2(30);

begin

    select uc.aciklama into v_ucret_duzey
    from personel pr, ucret_duzey uc
    where personel_id = p_id
        and pr.maas between uc.maas_alt_limit and uc.maas_ust_limit;

    return v_ucret_duzey;

end;

-----------------------------------------
--Fonksiyonları SQL İçinde Kullanma
-----------------------------------------

select ad, soyad, maas,
       f_ucret_duzeyi(personel_id) ucret_duzeyi
from personel

-----------------------------------------

select ucret_duzeyi, count(*) adet from
(
    select f_ucret_duzeyi(personel_id) ucret_duzeyi
    from personel
)
group by ucret_duzeyi
```

```
-----------------------------------------

create or replace function f_kesinti
(p_id personel.personel_id%type)
return number
is
    v_kesinti number;

begin

    select decode(unvan,
        'UZMAN', 0.05,
        'MÜDÜR', 0.08,
        'GRUP MÜDÜRÜ', 0.20,
        0) * maas into v_kesinti
    from personel
    where personel_id = p_id;

    return v_kesinti;

end;

select ad, soyad, unvan,
       f_kesinti(personel_id) kesinti
from personel
order by 4 desc;

select dept_id,
   max(f_kesinti(personel_id)) maks_kesinti
from personel
group by dept_id

-----------------------------------------
--Fonksiyonlar - Örnek
-----------------------------------------

create or replace function f_date_diff
(
    p_sure_tipi in varchar2,
    p_d1        in date,
    p_d2        in date
)
return number
as
    v_sonuc    number;

--  p_sure_tipi değeri=> ss : Saniye, mi : Dakika, hh : Saat

begin

    select (p_d2 - p_d1) *
            decode( upper(p_sure_tipi),
                    'SS', 24*60*60,
                    'MI', 24*60,
                    'HH', 24,
                    null )
    into v_sonuc from dual;

    return v_sonuc;
end;
```

```
-----------------------------------------

create or replace function f_zam_orani_hesapla(p_id number) return number
is
    v_zam_orani number;
    v_unvan varchar2(20);

begin

    select unvan into v_unvan from personel
    where personel_id = p_id;

    case v_unvan
        when 'UZMAN' then v_zam_orani := 1.05;
        when 'MÜDÜR' then v_zam_orani := 1.10;
        when 'TEKNİKER' then v_zam_orani := 1.07;
        else
            raise_application_error(-20001, 'Bu unvana ait zam oranı bulunamadı');
    end case;

    return v_zam_orani;
    exception
        when no_data_found then
            raise_application_error(-20002, p_id || ' numaralı personel bulunamadı!!');
            return null;

end;

-----------------------------------------
--Fonksiyonlar – Result Cache
-----------------------------------------

create or replace function f_bilgi_rc(p_id number)
return varchar2
result_cache
is
    v_ad varchar2(40);
begin

    select ad into v_ad
    from personel
    where personel_id = p_id;

    dbms_output.put_line(p_id|| ': '||v_ad);
    return v_ad;

end;

-----------------------------------------

declare
    type t_sicil is table of number;
    v_sicil t_sicil;
    v_cikti varchar2(50);

begin
    v_sicil := t_sicil(5010, 5020, 5030, 5010, 5050);
    for i in 1..v_sicil.count loop

        v_cikti := f_bilgi_rc(v_sicil(i));
        dbms_output.put_line(v_cikti);

    end loop;
end;
```

```
-------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------

-----------------------------------------
--Prosedürler
-----------------------------------------

create or replace procedure print_grand_lux_products
is

     cursor c_product is
        select name, price, price * discount as discount,
            price - price * discount as net_price
        from product p, product_segment ps
        where p.segment_id = ps.id
            and ps.segment = 'Grand Luxury';

begin

    for row_product in c_product loop

        dbms_output.put_line(row_product.name ||' Price: '||
                             row_product.price || ' Discount: '||
                             row_product.discount|| ' Net Price: '||
                             row_product.net_price);
    end loop;

end;

-----------------------------------------
--Prosedürler - Parametreli
-----------------------------------------

create or replace procedure top_ten_orders(p_ship_via number)
is
    cursor c_order_info is
        select * from
        (
            select first_name, last_name, freight
            from orders o, employees e
            where o.ship_via = p_ship_via
                and o.employee_id = e.employee_id
            order by freight desc
        )
        where rownum < 11;

begin

    for v_order_info in c_order_info loop

        dbms_output.put_line(v_order_info.first_name || ' ' ||
            v_order_info.last_name || ': ' ||
            v_order_info.freight);

    end loop;

end;
```

```
-------------------------------------------
--Fonksiyonları SQL İçinde Kullanma
-------------------------------------------

create or replace function get_manager (p_emp_id number)
return varchar2
result_cache
is
    v_manager_name varchar2(100);

begin

    select first_name || ' ' || last_name
        into v_manager_name
    from employees
    where employee_id = p_emp_id;

    return v_manager_name;

end;

-------------------------------------------
--Fonksiyonlar
-------------------------------------------

create or replace function get_total_orders(
    p_year pls_integer
)
return number
is
    v_total_orders number := 0;

begin
    -- get total sales
    select sum(unit_price * quantity)
        into v_total_orders
    from order_details
        inner join orders using (order_id)
    where shipped_date is not null
    group by extract(year from order_date)
    having extract(year from order_date) = p_year;

    return v_total_orders;
end;
```

```
-------------------------------------------
--Paket Oluşturma
-------------------------------------------

create or replace package pck_genel as

end pck_genel;

---------

create or replace package body pck_genel as

end pck_genel;

-------------------------------------------
--Paket İçerisine Altprogram Ekleme
-------------------------------------------

--Body

create or replace package body pck_genel as

function date_diff(p_sure varchar2,p_d1 date, p_d2 date)
return number
as
    v_sonuc    number;
begin

select (p_d2 - p_d1) *
            decode( upper(p_sure),
                    'SS', 24*60*60, 'MI', 24*60, 'HH', 24, null )
    into v_sonuc from dual;

    return v_sonuc;

end;

end pck_genel;

--Spec

create or replace package pck_genel as

function date_diff(p_sure varchar2,p_d1 date, p_d2 date)
return number;

end pck_genel;
```

```
-------------------------------------------
--Paket Altprogramlarını Çağırma
-------------------------------------------

declare
    v_sure varchar2(20);
begin

    v_sure := pck_genel.date_diff('hh', sysdate-10, sysdate);
    dbms_output.put_line(v_sure);

end;

-------------------------------------------

begin

    dbms_output.put_line(pck_genel.date_diff('hh', sysdate-10, sysdate));

end;

-------------------------------------------

select pck_genel.date_diff('hh', sysdate-10, sysdate) sure from dual;

-------------------------------------------
--Paket İçerisine Altprogram Ekleme
-------------------------------------------

…
procedure out_yaz(p_deger varchar2)
is
begin
    dbms_output.put_line(p_deger);
end;

…

procedure maas_guncelle(p_id number, p_yeni_maas number)
is
begin

    update personel set maas = p_yeni_maas
    where personel_id = p_id;

    out_yaz('Güncellenen kayıt sayısı: ' || sql%rowcount);

end;
…
```

```
-------------------------------------------
--Body Bölümü Olmayan Paketler
-------------------------------------------

create or replace package pck_sabitler as

    c_mil2metre  constant number := 1609.3;
    c_fit2metre  constant number := 0.3048;
    c_metre2fit  constant number := 3.28;
    c_cm2inc     constant number := 0.39;

end pck_sabitler;

…

set serveroutput on;
begin

    dbms_output.put_line('50 Mil '|| 50 * pck_sabitler.c_mil2metre || ' metredir');

end;

-------------------------------------------
--Paketlerdeki Global Değişkenler
-------------------------------------------

create or replace package body pck_genel as

    v_global_number number := 0;

procedure set_global(p_deger number)
is
begin
    v_global_number := p_deger;
end;
…
…

set serveroutput on;
begin

    dbms_output.put_line(pck_genel.v_global_number);

    pck_genel.v_global_number := 100;

    dbms_output.put_line(pck_genel.v_global_number);

    pck_genel.set_global(20);

    dbms_output.put_line(pck_genel.v_global_number);

end;

-------------------------------------------
--Paketlerin Kodları Nerede Saklanıyor?
-------------------------------------------

select * from user_source
where name = 'TELNO_FORMATLA'
order by line;

select * from user_source
where name = 'PCK_GENEL'
      and type = 'PACKAGE BODY'
order by line;
```

```
-------------------------------------------
--Paketlerde Overloading
-------------------------------------------

…
procedure konum_ekle(p_konum_adi varchar2) is
    max_id integer;
begin

    select max(konum_id)+1 into max_id from konum;
    insert into konum values(max_id, p_konum_adi, 34);
    commit;

end;

procedure konum_ekle(p_konum_adi varchar2, p_il_kodu integer) is
    max_id integer;
begin

    select max(konum_id)+1 into max_id from konum;
    insert into konum values(max_id, p_konum_adi, p_il_kodu);
    commit;

end;
…

-------------------------------------------

function date_diff(p_sure varchar2, p_d1 date, p_d2 date) return number as
    v_sonuc    number;
begin
    select (p_d2 - p_d1) *
            decode( upper(p_sure), 'SS', 24*60*60, 'MI', 24*60, 'HH', 24, null )
    into v_sonuc from dual;
    return v_sonuc;
end;

function date_diff(p_d1 date, p_d2 date) return number as
    v_sonuc    number;
    p_sure varchar2(2);
begin
    p_sure := 'ss';
    select (p_d2 - p_d1) *
            decode( upper(p_sure), 'SS', 24*60*60, 'MI', 24*60, 'HH', 24, null )
    into v_sonuc from dual;
    return v_sonuc;
end;

function date_diff(p_d1 date) return number as
    v_sonuc    number;
    p_sure varchar2(2) := 'ss';
    p_d2 date := sysdate;
begin
    select (p_d2 - p_d1) *
            decode( upper(p_sure), 'SS', 24*60*60, 'MI', 24*60, 'HH', 24, null )
    into v_sonuc from dual;
    return v_sonuc;
end;
```

```
-------------------------------------------
--Serially Reusable Paketler
-------------------------------------------

create or replace package not_serially_reusable_pkg as
  v_not_sr int := 0;
end;

-------------

create or replace package serially_reusable_pkg as
  pragma serially_reusable;
  v_sr int := 0;
end;

begin
    not_serially_reusable_pkg.v_not_sr := 100;
    serially_reusable_pkg.v_sr := 100;
end;

-------------

begin
    dbms_output.put_line ('not_serially: ' || not_serially_reusable_pkg.v_not_sr );
    dbms_output.put_line ('serially: ' || serially_reusable_pkg.v_sr );
end;

-------------

begin
    not_serially_reusable_pkg.v_not_sr := 100;
    serially_reusable_pkg.v_sr := 100;
    dbms_output.put_line ('not_serially: ' || not_serially_reusable_pkg.v_not_sr );
    dbms_output.put_line ('serially: ' || serially_reusable_pkg.v_sr );
end;
```

```
-------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------

-----------------------------------------
--Paket İçerisine Altprogram Ekleme
-----------------------------------------

create or replace package pck_genel as

function yonetici_getir (p_personel_id number) return varchar2;

end pck_genel;

/

create or replace package body pck_genel as

    function yonetici_getir (p_personel_id number) return varchar2
    as
        v_yonetici_ismi varchar2(100);
    begin

        select
             pry.ad || ' ' ||pry.soyad
             into v_yonetici_ismi
        from personel pr, yonetici yn, personel pry
        where pr.yonetici_id = yn.yonetici_id
              and pry.personel_id = yn.personel_id
              and pr.personel_id = p_personel_id;

        return v_yonetici_ismi;

        exception
            when no_data_found then
                return 'Yönetici bulunamadı';

    end;

end pck_genel;
```

```
-----------------------------------------
--Paketlerdeki Global Değişkenler
-----------------------------------------

create or replace package pck_order_report as

procedure find_good_companys;

end pck_order_report;

/

create or replace package body pck_order_report as

    --global değişkenler tanımlanıyor
    vgb_category_id pls_integer;
    type tgb_order_id_type is varray(3) of integer;
    vgb_order_ids tgb_order_id_type := tgb_order_id_type(null, null, null);

procedure find_max_category
is
begin

    --stock ücreti en büyük miktara
    --sahip kategory tespit ediliyor
    select category_id into vgb_category_id from
    (
        select category_id,
            unit_price*units_in_stock as stock_price
        from products
        order by 2 desc
    )
    where rownum = 1;
    dbms_output.put_line('Category id: '||vgb_category_id);

end;

procedure find_orders
is
    --Yukarıda bulunan kategoideki en yüksek satış miktarlı
    --ilk 3 sipariş bulunuyor
    cursor crs_orders is
        select rownum, order_id from
        (
            select order_id, unit_price*quantity
            from order_details od
            where od.product_id in
                (
                    select product_id from products p
                    where category_id = vgb_category_id
                )
            order by 2 desc
        )
        where rownum < 4;
begin

    --Bulunan her bir sipariş id değer, global
    --bir varray dizisinin bir elemanına atanıyor
    for crs_row in crs_orders loop
        vgb_order_ids(crs_row.rownum) := crs_row.order_id;
        dbms_output.put_line('Order id: '||crs_row.order_id);
    end loop;

end;
```

```
procedure find_companys
is
    --Bulunan herbir siparişi veren şirketler bulunuyor
    --parmaetreli cursor kullanılıyor
    cursor crs_company(v_order_id integer) is
        select company_name from customers c
        where customer_id in
        (
            select customer_id from orders
            where order_id = v_order_id
        );
begin

    for i in 1..vgb_order_ids.count loop

        --Her bir sipariş id, cusror'e parametre olarak gönderiliyor
        for crs_row in crs_company(vgb_order_ids(i)) loop

            dbms_output.put_line('Company name: '|| crs_row.company_name);

        end loop;

    end loop;

end;

procedure find_good_companys
is
begin
    --Dışarıdan çağrılacak ana fonksiyon.
    --Diğer fonksiyonları sırayla çağırıyor
    find_max_category;
    find_orders;
    find_companys;
end;

end pck_order_report;
```

```
-------------------------------------------
--DBMS_OUTPUT
-------------------------------------------

set serveroutput on;
begin

    dbms_output.put_line('put_line');
    dbms_output.put('put');
    dbms_output.new_line;
    dbms_output.put_line('put_line');

end;

-------------------------------------------

declare
   lines dbms_output.chararr;
   num_lines number := 0;

begin

   -- enable the buffer with default size 20000
   dbms_output.enable;

   dbms_output.put_line('Merhaba!');
   dbms_output.put_line('İkinci satırı ekliyoruz!');
   dbms_output.put_line('Üçüncü satırı ekliyoruz!');

   dbms_output.get_lines(lines, num_lines);

   for i in 1..num_lines loop

      dbms_output.put_line(i||'. satır: '||lines(i));

   end loop;
end;

-------------------------------------------
--Oracle Dizini Oluşturma
-------------------------------------------

$ docker exec -it db_oracle_registry bash

bash-4.2$ mkdir egitim_dir

bash-4.2$ pwd

CREATE OR REPLACE DIRECTORY EGITIM_DIR AS
'C:\Oracle\product\12.2.0\Egitim_dir';

GRANT READ, WRITE ON DIRECTORY EGITIM_DIR TO EGITIM;
```

```
-------------------------------------------
--UTL_FILE: Dosya Yazma
-------------------------------------------

procedure dosya_yaz(p_dosya_ismi varchar2, p_dizin varchar2)
is
    filex utl_file.file_type;

begin

    filex := utl_file.fopen(p_dizin, p_dosya_ismi, 'W');
    utl_file.put_line(filex, 'Merhaba');
    utl_file.put_line(filex, 'Bu dosya PL/SQL eğitimi için oluşturuldu');
    utl_file.put_line(filex, 'UTL_FILE paketi dosya okuma/yazma');
    utl_file.fclose(filex);

end;

-------------------------------------------
--UTL_FILE: Dosya Okuma
-------------------------------------------

procedure dosya_oku(p_dosya_ismi varchar2, p_dizin varchar2)
is
    filex utl_file.file_type;
    satir varchar2(150);
    satir_sayisi pls_integer := 1;
begin
    if not utl_file.is_open(filex) then
        filex := utl_file.fopen(p_dizin, p_dosya_ismi, 'R');

        begin
            loop
                utl_file.get_line(filex, satir);
                dbms_output.put_line(satir_sayisi || '. satır: ' || satir);
                satir_sayisi := satir_sayisi + 1;
            end loop;
            exception when no_data_found then
                dbms_output.put_line('--- Dosya Sonu ---');
        end;
        dbms_output.put_line('Toplam: '|| satir_sayisi ||' satır okundu');
        utl_file.fclose(filex);
    end if;
end;

-------------------------------------------
--UTL_MAIL - Örnek-1
-------------------------------------------

--connect as sysdba ve utl_mail paketi kurulur
@ORACLE_HOME/rdbms/admin/utlmail.sql
@ORACLE_HOME/rdbms/admin/prvtmail.sql

--smtp parametresi ayarlanır
ALTER SYSTEM SET SMTP_OUT_SERVER='smtp.server.com' SCOPE=SPFILE;

--mail göndermek için örnek
begin
    utl_mail.send('tuncay@oracle.com', 'egitim@oracle.com'
       message=> 'PL/SQL eğitim notlarını iyi alınız'
       subject=> 'Eğitim notları');
end;
```

```
-------------------------------------------
--UTL_MAIL - Örnek-2
-------------------------------------------

declare
    l_attachment    raw(32767);

begin

    select rawimage
      into l_attachment
      from my_images
    where id = 1;

    utl_mail.send_attach_raw
    (
      sender       => 'me@domain.com',
      recipients   => 'person1@domain.com,person2@domain.com',
      subject      => 'UTL_MAIL Test',
      message      => 'Mail testi yapılıyor, dikkate almayın!',
      attachment   => l_attachment, --resim_getir('sql.gif', dizin)
      att_filename => 'clouds.jpg'
    );

exception
    when others then
        raise_application_error(-20001,'Beklenmeyen bir hata oluştu: ' || sqlerrm);
end;

-------------------------------------------
--UTL_MAIL - RAW Dosya Oluşturma
-------------------------------------------

create or replace function resim_getir
(
    p_dosya_ismi varchar2,
    p_dizin varchar2
)
return raw is

    resim raw(32767);
    dosya BFILE := BFILENAME(p_dizin, p_dosya_ismi);

begin

    DBMS_LOB.FILEOPEN(dosya, DBMS_LOB.FILE_READONLY);
    resim := DBMS_LOB.SUBSTR(dosya);
    DBMS_LOB.CLOSE(dosya);
    return resim;

end;
```

```
-----------------------------------------
--UTL_MAIL – Text Dosya Oluşturma
-----------------------------------------

create or replace function text_getir
(
    p_dosya_ismi varchar2,
    p_dizin varchar2
)
return varchar2 is

    icerik varchar2(32767);
    dosya BFILE := BFILENAME(p_dizin, p_dosya_ismi);

begin

    DBMS_LOB.FILEOPEN(dosya, DBMS_LOB.FILE_READONLY);
    icerik := UTL_RAW.CAST_TO_VARCHAR2(DBMS_LOB.SUBSTR(dosya));
    DBMS_LOB.CLOSE(dosya);
    return icerik;

end;
```

```
-------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------

-----------------------------------------
--UTL_FILE: Dosya Yazma
-----------------------------------------

create or replace procedure write_student_info(p_dir varchar2, p_file_name varchar2)
is
    filex utl_file.file_type;

begin

    filex := utl_file.fopen(p_dir, p_file_name, 'W');

    for std_row in (select * from student) loop

        utl_file.put_line(filex,
            'Name: ' || std_row.name ||
            ', Course: ' || std_row.course_name ||
            ', Class: ' || std_row.class_no);

    end loop;

    utl_file.fclose(filex);

end;

exec write_student_info('EGITIM_DIR', 'student_info.txt');
```

```
-----------------------------------------
--UTL_FILE: Dosya Okuma
-----------------------------------------

create or replace procedure read_student_info(p_dir varchar2, p_file_name varchar2)
is
    filex utl_file.file_type;
    row_text varchar2(300);
    row_count pls_integer := 0;
    sci_count pls_integer := 0;
begin

    if not utl_file.is_open(filex) then
        filex := utl_file.fopen(p_dir, p_file_name, 'R');

        begin
            loop
                utl_file.get_line(filex, row_text);

                if instr(lower(row_text), 'science') > 0 then
                    dbms_output.put_line(row_count || '. ' || row_text);
                    sci_count := sci_count + 1;
                end if;

                row_count := row_count + 1;

            end loop;

            exception
                when no_data_found then
                    dbms_output.put_line('--- End of file ---');

        end;
        dbms_output.put_line('Sum of rows: '|| row_count);
        dbms_output.put_line('Sum of science: '|| sci_count);
        utl_file.fclose(filex);

    end if;

    exception
    when utl_file.invalid_operation then
        dbms_output.put_line('File not found');
end;
```

```
-------------------------------------------
--EXECUTE IMMEDIATE - Sorgu
-------------------------------------------

declare
    v_ad varchar2(50);
    v_maas number;

begin

    execute immediate
      'select ad, maas from personel where personel_id = :b1'
        into v_ad, v_maas using 5020;

    dbms_output.put_line(v_ad || ': '||v_maas);

end;

-------------------------------------------

function personel_getir(p_id number) return personel%rowtype
is
    v_sql varchar2(500) := 'select * from personel where personel_id = :b1';
    v_persrow personel%rowtype;
begin

    execute immediate v_sql into v_persrow using p_id;
    return v_persrow;

end;

declare
    v_persrow personel%rowtype;
begin

    v_persrow := PCK_GENEL.PERSONEL_GETIR(5015);
    dbms_output.put_line(v_persrow.ad || ', ' || v_persrow.unvan);

end;

-------------------------------------------
--EXECUTE IMMEDIATE - DML
-------------------------------------------

begin

    execute immediate
      'update personel set maas = maas * :p1 where unvan = :b2'
        using 1.15, 'UZMAN';

    dbms_output.put_line(sql%rowcount);

end;

-------------------------------------------

begin

    execute immediate 'insert into departman values(:1, :2)'
        using 115, 'Dijital dönüşüm Ofisi';

end;
```

```
-------------------------------------------
--EXECUTE IMMEDIATE - DDL
-------------------------------------------

begin

    execute immediate 'create table is_ilanlari (id number)';

    execute immediate 'alter table is_ilanlari add baslik varchar2(25)';

    execute immediate 'truncate table is_ilanlari';

end;

-------------------------------------------
--Data Dictionary ile Dinamik SQL Yazma
-------------------------------------------

SELECT
    'ALTER TABLE ' || TABLE_NAME ||
    ' RENAME TO '  || 'T_' ||TABLE_NAME||';' AS SCRIPT
FROM USER_TABLES;

-------------------------------------------

SELECT 'ALTER TABLE ' || TABLE_NAME ||
    ' ADD ID NUMBER;' AS SCRIPT FROM
(
    SELECT TABLE_NAME FROM USER_TABLES
    MINUS
    SELECT TABLE_NAME FROM USER_TAB_COLUMNS
    WHERE COLUMN_NAME = 'ID'
)

-------------------------------------------
--EXECUTE IMMEDIATE - Dinamik PL/SQL
-------------------------------------------

function yillik_maas(p_id number) return number
is
    v_plsql varchar2(500) :=
        'declare ' ||
        'v_persrow personel%rowtype; ' ||
        'begin ' ||
        'v_persrow := pck_genel.personel_getir(:persid); ' ||
        ':sonuc := v_persrow.maas * 12; ' ||
        'end;';
    v_sonuc number;
begin

    dbms_output.put_line(v_plsql);
    execute immediate v_plsql using in p_id, out v_sonuc;
    return v_sonuc;

end;

exec dbms_output.put_line(pck_genel.yillik_maas(5015));
```

```
-----------------------------------------
--BULK COLLECT INTO
-----------------------------------------

set serveroutput on;
declare
    type t_ad   is table of varchar2 (20);
    type t_maas is table of number;

    v_ad   t_ad;
    v_maas t_maas;
Begin

    select ad, maas
        bulk collect into v_ad, v_maas
    from personel;

    for idx in 1..v_ad.count
    loop
        dbms_output.put_line (idx||' - '||v_ad (idx) ||': '||v_maas (idx));
    end loop;

end;

-----------------------------------------
--BULK COLLECT INTO - Limit
-----------------------------------------

declare
  cursor cur_personel is
     select * from personel;

    type t_personel is table of cur_personel%rowtype;

    v_per_dizi t_personel;
begin
    open cur_personel;

        fetch cur_personel
            bulk collect into v_per_dizi limit 50;

        for indx in 1 ..v_per_dizi.count
        loop
            dbms_output.put_line('Adı:' || v_per_dizi(indx).ad ||
                                 ' Soyadı:' || v_per_dizi(indx).soyad);
        end loop;

   close cur_personel;
end;
```

```
-----------------------------------------
--BULK COLLECT INTO - FORALL
-----------------------------------------

declare
  cursor cur_personel is
     select * from personel;

    type t_personel is table of cur_personel%rowtype;

    v_per_dizi t_personel;
begin
    open cur_personel;

        fetch cur_personel
            bulk collect into v_per_dizi limit 10;

        forall indx in v_per_dizi.first..v_per_dizi.last
            update personel set prim=111 where personel_id = v_per_dizi(indx).personel_id;

   close cur_personel;
end;

-----------------------------------------
--OPEN FOR İfadesi
-----------------------------------------

set serveroutput on;
declare
    type t_refc is ref cursor;
    type t_ad   is table of varchar2 (20);
    type t_maas is table of number;

    c_pers t_refc;
    v_ad   t_ad;
    v_maas t_maas;

begin

    open c_pers for 'select ad, maas from personel';
    fetch c_pers bulk collect into v_ad, v_maas;
    close c_pers;

    for idx in 1..v_ad.count
    loop
        dbms_output.put_line (idx||' - '||v_ad (idx) ||': '||v_maas (idx));
    end loop;

end;
```

```
-----------------------------------------
--DBMS_SQL Örnek-1
-----------------------------------------

function tum_kayitlari_sil(p_tablo_ismi varchar2) return number
is
    v_cur_id pls_integer;
    v_del_rows number;
    v_sql varchar2(100);

begin

    v_sql := 'delete from ' || p_tablo_ismi;
    v_cur_id := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(v_cur_id, v_sql, DBMS_SQL.NATIVE);
    v_del_rows := DBMS_SQL.EXECUTE(v_cur_id);
    DBMS_SQL.CLOSE_CURSOR(v_cur_id);
    return v_del_rows;

end;

--create table departman_temp as select * from departman;

declare
    v_tablo varchar2(20) := 'DEPARTMAN_TEMP';
    v_silinen_kayit integer := 0;

begin

    v_silinen_kayit := pck_genel.tum_kayitlari_sil(v_tablo);
    dbms_output.put_line( v_tablo || ' tablosundan ' || v_silinen_kayit ||
    ' adet kayıt silinmiştir');

end;
```

```
-------------------------------------------
--DBMS_SQL Örnek-2
-------------------------------------------

procedure is_ilani_ekle(
    p_ilan_id integer,
    p_baslik varchar2,
    p_tarih date,
    p_platform varchar2 default 'Linkedin')
is
    v_cur_id pls_integer;
    v_sql varchar2(100);
    v_rows integer;
begin

    v_sql := 'insert into is_ilanlari values (:bid, :bbaslik, :btarih, :bplatform)';
    v_cur_id := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(v_cur_id, v_sql, DBMS_SQL.NATIVE);
    DBMS_SQL.BIND_VARIABLE(v_cur_id, ':bid', p_ilan_id);
    DBMS_SQL.BIND_VARIABLE(v_cur_id, ':bbaslik', p_baslik);
    DBMS_SQL.BIND_VARIABLE(v_cur_id, ':btarih', p_tarih);
    DBMS_SQL.BIND_VARIABLE(v_cur_id, ':bplatform', p_platform);
    v_rows := DBMS_SQL.EXECUTE(v_cur_id);
    DBMS_SQL.CLOSE_CURSOR(v_cur_id);

end;

--create sequence sq_is_ilani start with 1 increment by 1;

begin
    PCK_GENEL.IS_ILANI_EKLE(1, 'PL/SQL Developer', sysdate+2, 'Kariyer');
    PCK_GENEL.IS_ILANI_EKLE(2, 'Oracle DBA', sysdate+1);
    PCK_GENEL.IS_ILANI_EKLE(sq_is_ilani.nextval, 'Senior Java Developer', sysdate+5);
end;

-------------------------------------------
--DBMS_SQL Örnek-3
-------------------------------------------

procedure departman_ekle is
    v_cur_id pls_integer;
    v_sql varchar2(100);
    v_rows integer;

    departid_array     DBMS_SQL.NUMBER_TABLE;
    deptname_array     DBMS_SQL.VARCHAR2_TABLE;
begin
    departid_array(1) := 116;
    departid_array(2) := 117;
    departid_array(3) := 118;

    deptname_array(1) := 'Uzay Bilimleri';
    deptname_array(2) := 'Yapay Zeka Ar-Ge';
    deptname_array(3) := 'Geri Dönüşüm Ar-Ge';

    v_sql := 'insert into departman values (:depid_array, :depname_array)';
    v_cur_id := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(v_cur_id, v_sql, DBMS_SQL.NATIVE);
    DBMS_SQL.BIND_ARRAY(v_cur_id, ':depid_array', departid_array);
    DBMS_SQL.BIND_ARRAY(v_cur_id, ':depname_array', deptname_array);
    v_rows := DBMS_SQL.EXECUTE(v_cur_id);
    DBMS_SQL.CLOSE_CURSOR(v_cur_id);
end;
```

```
-----------------------------------------
--DBMS_SQL Örnek-4
-----------------------------------------

 procedure personel_yazdir is
        v_cur_id pls_integer;
        v_sql varchar2(100);
        v_rows integer;

        col_ad  varchar2(30);
        col_maas number;
    begin

      v_sql := 'Select ad, maas from personel where unvan=''UZMAN''';
      v_cur_id := DBMS_SQL.OPEN_CURSOR;
      DBMS_SQL.PARSE(v_cur_id, v_sql, DBMS_SQL.NATIVE);
      v_rows := DBMS_SQL.EXECUTE(v_cur_id);

      DBMS_SQL.DEFINE_COLUMN(v_cur_id, 1, col_ad, 30);
      DBMS_SQL.DEFINE_COLUMN(v_cur_id, 2, col_maas);

      while DBMS_SQL.FETCH_ROWS(v_cur_id) > 0 loop

        DBMS_SQL.COLUMN_VALUE(v_cur_id, 1, col_ad);
        DBMS_SQL.COLUMN_VALUE(v_cur_id, 2, col_maas);
        DBMS_OUTPUT.PUT_LINE(col_ad || ' : ' || col_maas);

      end loop;
      DBMS_SQL.CLOSE_CURSOR(v_cur_id);
    end;
```

```
------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
------------------------------------------------

-----------------------------------------
--EXECUTE IMMEDIATE - Sorgu
-----------------------------------------

declare
    v_sql varchar2(500) := 'select fruit_a from basket_a where id_a = :b1';
    v_fruit varchar2(25);

begin

    for i in 1..5 loop

        execute immediate v_sql into v_fruit using i;
        dbms_output.put_line(v_fruit);

    end loop;

end;

-----------------------------------------
--EXECUTE IMMEDIATE - DML
-----------------------------------------

declare
    type cars_type is table of varchar2(30);

    v_car_brand cars_type := cars_type('Bugatti', 'McLaren', 'Lamborghini');
    v_car_price cars_type := cars_type('200000', '250000', '300000');

    v_sql varchar2(100) := 'insert into cars(id, brand, price) values(:b1, :b2, :b3)';
    v_max_id pls_integer;

begin

    select max(id) into v_max_id from cars;

    for i in v_car_brand.first..v_car_brand.last loop

        execute immediate v_sql using v_max_id + i, v_car_brand(i), to_number(v_car_price(i));

    end loop;
end;


-----------------------------------------
--Data Dictionary ile Dinamik SQL Yazma
-----------------------------------------

SELECT
    'ALTER TABLE '||TABLE_NAME||
    ' DISABLE CONSTRAINT '||CONSTRAINT_NAME||';'
FROM USER_CONSTRAINTS;
```

```
-----------------------------------------
--BULK COLLECT INTO - Limit
-----------------------------------------

declare
    cursor crs_stock is
        select product_name, company_name,
            sum(units_in_stock) stock_amount
        from products p, suppliers s
        where p.supplier_id = s.supplier_id
        group by product_name, company_name
        order by 3 desc;

    type t_stock is table of crs_stock%rowtype;
    v_stock_info t_stock;

begin

    open crs_stock;

        fetch crs_stock
            bulk collect into v_stock_info limit 10;

        for i in 1 ..v_stock_info.count
        loop
            dbms_output.put_line('[Product Name:' || v_stock_info(i).product_name ||
                                 '] [Sup. Comp. Name:' || v_stock_info(i).company_name ||
                                 '] [Stock Amunt:' || v_stock_info(i).stock_amount ||']');
        end loop;

   close crs_stock;

end;
```

```
-----------------------------------------
--BULK COLLECT INTO - FORALL
-----------------------------------------

--alter table order_details add last_price number;
--alter table order_details add order_details_id number;
--update order_details set order_details_id=rownum;

declare
  cursor crs_order_details is
     select * from order_details;

    type t_order_details is table of crs_order_details%rowtype;
    v_order_details t_order_details;

    v_time_start number;
    v_time_end number;

begin

    v_time_start := DBMS_UTILITY.get_time;

    for order_detail_row in crs_order_details loop
        update order_details set last_price = round(unit_price*quantity*(100-discount)/100,2)
        where order_details_id = order_detail_row.order_details_id;
    end loop;

    v_time_end := DBMS_UTILITY.get_time;
    dbms_output.put_line('For loop inserts: ' || (v_time_end - v_time_start));

    v_time_start := DBMS_UTILITY.get_time;
    open crs_order_details;

        fetch crs_order_details
            bulk collect into v_order_details;

        forall i in v_order_details.first..v_order_details.last
            update order_details set last_price = round(unit_price*quantity*(100-
discount)/100,2)
            where order_details_id = v_order_details(i).order_details_id;

    close crs_order_details;
    v_time_end := DBMS_UTILITY.get_time;
    dbms_output.put_line('Forall inserts: ' || (v_time_end - v_time_start));

   commit;

end;
```

```
-----------------------------------------
--DBMS_SQL
-----------------------------------------

create or replace function create_email(p_student_name varchar2, p_course_name varchar2)
return varchar2
is
    v_return_value  varchar2(500);
begin
    select
        lower(replace(p_student_name,' ','.'))||'@'||
        lower(replace(p_course_name,' ','.'))||'.com'
        into v_return_value
    from dual;

    return v_return_value;
end;

create or replace procedure update_email(p_course_name varchar2)
is
    v_cur_id pls_integer;
    v_up_rows number;
    v_sql varchar2(100);

begin

    v_sql := 'update student set email = create_email(name, course_name) where course_name =
:cn';
    v_cur_id := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(v_cur_id, v_sql, DBMS_SQL.NATIVE);
    DBMS_SQL.BIND_VARIABLE(v_cur_id, ':cn', p_course_name);
    v_up_rows := DBMS_SQL.EXECUTE(v_cur_id);
    DBMS_SQL.CLOSE_CURSOR(v_cur_id);
    dbms_output.put_line('Updated rows:' || v_up_rows);
    commit;
end;
```

```
-------------------------------------------
--Trigger Oluşturma - DML
-------------------------------------------

create or replace trigger trg_dep_mesai
before insert
on departman
begin

    if to_char(sysdate, 'D') in (7,1) then
        raise_application_error(-20001,
    'Hafta sonu veri girişi yapılmamalıdır');
    end if;

end;

-------------------------------------------

create or replace trigger trg_dep_mesai
before insert or update or delete
on departman
begin

    if to_char(sysdate, 'D') in (7,1) then
        if INSERTING then
            raise_application_error(-20001, 'Hafta sonu veri girişi yapılmamalıdır');
        elsif DELETING then
            raise_application_error(-20002, 'Hafta sonu veri silmesi yapılmamalıdır');
        elsif UPDATING ('DEPT_ID') then
            raise_application_error(-20003, 'Hafta sonu veri güncellemesi yapılmamalıdır');
        end if;
    end if;

end;
-------------------------------------------

insert into departman values(150, 'Yeni ofis');

update departman set dept_ismi = 'Sakarya Yeni Ofis'
where dept_id = 100;

delete departman where dept_id = 100;

-------------------------------------------
--Row Level Trigger
-------------------------------------------

CREATE OR REPLACE TRIGGER trg_per_maas_kontrol
before insert or update on personel
for each row
declare
    maas_ok boolean := true;
begin

    case :new.dept_id
         when 110 then if :new.maas > 10000 then maas_ok := false; end if;
         when 100 then if :new.maas > 12000 then maas_ok := false; end if;
    end case;

    if not maas_ok then
       raise_application_error(-20002, 'Maaş değerini yüksek girdiniz');
    end if;

end;
```

```
-------------------------------------------
--Trigger Oluşturma – NEW, OLD Yapısı
-------------------------------------------

create or replace trigger trg_per_maas
after update on personel
--REFERENCING NEW AS NEW OLD AS OLD
for each row
declare
    v_maas_farki number;
begin

    v_maas_farki := :new.maas - :old.maas;
    dbms_output.put_line('Eski Maaş:'  || :old.maas);
    dbms_output.put_line('Yeni Maaş:'  || :new.maas);
    dbms_output.put_line('Maaş Farkı:' || v_maas_farki);

end;

-------------------------------------------

CREATE OR REPLACE TRIGGER TRG_KON_ILLER
BEFORE INSERT OR UPDATE ON KONUM
REFERENCING NEW AS yeni OLD AS eski
FOR EACH ROW
declare
    v_adet integer;
begin

    select count(*) into v_adet from iller
    where il_kodu = :yeni.il_kodu;

    if v_adet = 0 then
       raise_application_error(-20001, 'Geçersiz il kodu girdiniz');
    end if;

end;


-------------------------------------------
--Trigger When İfadesi
-------------------------------------------

create or replace trigger trg_per_maas_kontrol
before insert or update on personel
for each row
WHEN (new.unvan = 'MÜHENDİS')
declare
    maas_ok boolean := true;
begin

    case :new.dept_id
         when 110 then if :new.maas > 10000 then maas_ok := false; end if;
         when 100 then if :new.maas > 12000 then maas_ok := false; end if;
    end case;

    if not maas_ok then
       raise_application_error(-20002, 'Maaş değerini yüksek girdiniz');
    end if;

end;
```

```
-------------------------------------------

create or replace trigger trg_is_ilani
  before insert or update
  on is_ilanlari
  for each row
  when (instr(lower(new.baslik),'developer') > 0)
begin
    :new.platform := 'Linkedin';
end;

-------------------------------------------
--Trigger İle Veri Aktarma
-------------------------------------------

create table personel_silinen as select * from personel where 1=2;

-------------------------------------------

create or replace trigger trg_per_silinen
before delete on personel
referencing new as new old as old
for each row
begin

    insert into personel_silinen
    values(:OLD.PERSONEL_ID, :OLD.AD, :OLD.SOYAD, :OLD.MAAS, :OLD.GIRIS_TARIHI,
           :OLD.CIKIS_TARIHI, :OLD.SEMT, :OLD.PRIM, :OLD.UNVAN, :OLD.IZIN_GUNU,
           :OLD.YONETICI_ID, :OLD.KONUM_ID, :OLD.DEPT_ID);

end;

-------------------------------------------
--Trigger İle Log Tutma
-------------------------------------------

create table departman_log
(
    islem_tipi       varchar2(20),
    islem_saati      date default sysdate,
    eski_dept_id     number,
    eski_dept_ismi   varchar2(100),
    yeni_dept_id     number,
    yeni_dept_ismi   varchar2(100),
    kullanici        varchar2(30)
);

create or replace trigger trg_dep_log
before insert or update or delete on departman
for each row
begin
  if INSERTING then
    insert into departman_log (islem_tipi,yeni_dept_id,yeni_dept_ismi,kullanici)
    values ('Insert', :new.dept_id, :new.dept_ismi, user);
  elsif DELETING then
    insert into departman_log (islem_tipi,eski_dept_id,eski_dept_ismi,kullanici)
    values ('Delete', :old.dept_id, :old.dept_ismi, user);
  elsif UPDATING then
    insert into departman_log
(islem_tipi,eski_dept_id,eski_dept_ismi,yeni_dept_id,yeni_dept_ismi,kullanici)
    values ('Update', :old.dept_id, :old.dept_ismi,
            :new.dept_id, :new.dept_ismi, user);
  end if;
end;
```

```
-------------------------------------------
--Trigger İçinde Prosedür Çağırma
-------------------------------------------

create or replace trigger trg_dep_log
before insert or update or delete on departman
referencing new as new old as old
for each row
declare
    v_islem varchar2(20);
begin

    if INSERTING then
        v_islem := 'Insert';
    elsif DELETING then
        v_islem := 'Delete';
    elsif UPDATING then
        v_islem := 'Update';
    end if;

    pck_genel.dep_log_yaz (v_islem, :old.dept_id, :old.dept_ismi,
    :new.dept_id, :new.dept_ismi);

end;

procedure dep_log_yaz(
    p_islem_tipi varchar2,
    p_eski_id integer,
    p_eski_isim varchar2,
    p_yeni_id integer,
    p_yeni_isim varchar2)
is
begin

    insert into departman_log
    values(p_islem_tipi, sysdate, p_eski_id, p_eski_isim,
           p_yeni_id, p_yeni_isim, user);

end;

-------------------------------------------
--Trigger İçinde Prosedür Çağırma - Alıştırma Script
-------------------------------------------

create table table_dml_log
(
    dml_type varchar2(1),
    dml_time date default sysdate,
    user_name varchar2(50) default user,
    table_name varchar2(50),
    column_name varchar2(50),
    column_old_value varchar2(250),
    column_new_value varchar2(250)
);
```

```
-------------------------------------------
--Trigger Instead OF
-------------------------------------------

create table personel_ozluk as
select personel_id, ad, soyad, giris_tarihi, cikis_tarihi,
       semt, izin_gunu, yonetici_id, konum_id, dept_id
from personel;

create table personel_ucret as
select personel_id, maas, nvl(prim,0) prim, unvan,
    trunc(dbms_random.value(5,20)) zam_orani
from personel;

create or replace view vw_personel as
select * from personel;

create or replace trigger trg_per_dagit
instead of insert or update or delete on vw_personel
for each row
begin

    if INSERTING then
        insert into personel_ozluk
        values(:new.personel_id, :new.ad, :new.soyad,
               :new.giris_tarihi, :new.cikis_tarihi,
               :new.semt, :new.izin_gunu, :new.yonetici_id,
               :new.konum_id, :new.dept_id);
    elsif DELETING then
        delete from personel_ozluk where personel_id = :old.personel_id;
        delete from personel_ucret where personel_id = :old.personel_id;
    elsif UPDATING('MAAS') or UPDATING('PRIM') then
        update personel_ucret set maas = :new.maas where personel_id = :old.personel_id;
    end if;

end;

-------------------------------------------
--Trigger Çalışma Sırası
-------------------------------------------

create or replace trigger trg_iller_follow1
before insert on iller
for each row
begin
  dbms_output.put_line('trg_iller_follow1 - Çalıştırıldı');
end;

-----------------------------------------

create or replace trigger trg_iller_follow2
before insert on iller
for each row
follows trg_iller_follow1
begin
  dbms_output.put_line('trg_iller_follow2 - Çalıştırıldı');
end;
```

```
-------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------

-----------------------------------------
--Trigger Oluşturma - DML
-----------------------------------------

create or replace trigger trg_product_segment
before insert
on product_segment
declare
    v_count pls_integer;
begin

    select count(*) into v_count from product_segment;

    if v_count = 3 then
        raise_application_error(-20001, 'There can be a maximum of 3 segments');
    end if;

end;

-----------------------------------------
--Trigger Oluşturma – NEW, OLD Yapısı
-----------------------------------------

create or replace trigger trg_cars
before insert or update on cars
referencing new as new old as old
for each row

begin

    if INSERTING or UPDATING then

        if :new.price not between 10000 and 2000000 then

            raise_application_error(-20001, 'Price is out of limits');

        end if;

        if :new.discount is null then

            raise_application_error(-20002, 'Discount can not be null');

        end if;

    end if;

end;
```

```
-----------------------------------------
--Trigger İçinde Prosedür Çağırma
-----------------------------------------

create or replace procedure write_dml_log
(
    p_dml_type varchar2,
    p_table_name varchar2,
    p_column_name varchar2,
    p_column_old_value varchar2,
    p_column_new_value varchar2
)
is

begin

    insert into table_dml_log(dml_type, table_name,
    column_name, column_old_value, column_new_value)
    values(p_dml_type, p_table_name, p_column_name,
    p_column_old_value, p_column_new_value);

end;

create or replace trigger trg_cars
before update on cars
referencing new as new old as old
for each row

begin

    if UPDATING then

        if :new.brand <> :old.brand then
            write_dml_log('U', 'CARS', 'BRAND', :old.brand, :new.brand);
        end if;
        if :new.price <> :old.price then
            write_dml_log('U', 'CARS', 'PRICE', :old.price, :new.price);
        end if;
        if :new.discount <> :old.discount then
            write_dml_log('U', 'CARS', 'DISCOUNT', :old.discount, :new.discount);
        end if;

    end if;

end;
```

```
-------------------------------------------
--Compound Trigger Oluşturma
-------------------------------------------

CREATE OR REPLACE TRIGGER TRG_PER_COMP
FOR INSERT OR UPDATE ON PERSONEL
COMPOUND TRIGGER

    BEFORE STATEMENT IS
    BEGIN
        dbms_output.put_line('Before Statement Çalıştı');
    END BEFORE STATEMENT;

    BEFORE EACH ROW IS
    BEGIN
        dbms_output.put_line('Before Each Row Çalıştı');
    END BEFORE EACH ROW;

    AFTER EACH ROW IS
    BEGIN
        dbms_output.put_line('After Each Row Çalıştı');
    END AFTER EACH ROW;

    AFTER STATEMENT IS
    BEGIN
        dbms_output.put_line('After Statement Çalıştı');
    END AFTER STATEMENT;

END;

-------------------------------------------

create or replace trigger trg_per_comp
for insert or update on personel
compound trigger

    type t_per_array is table of number
        index by binary_integer;
    v_pers t_per_array;
    v_islem varchar2(1);

    after each row is
    begin
        v_pers(v_pers.count+1) := :new.personel_id;
    end after each row;

    after statement is
    begin
        if INSERTING then v_islem := 'I';
        else v_islem := 'U'; end if;
        forall i in 1 .. v_pers.count
            insert into per_log
            values (systimestamp, v_islem, v_pers(i));
    end after statement;

end;

create table per_log
(
    zaman        timestamp not null,
    islem_tipi   varchar2(1) not null,
    personel_id  integer not null
);
```

```sql
insert into personel
(personel_id, ad, soyad, maas)
values(6002, 'İsmet', 'Hayran', 8000);

insert into personel
(personel_id, ad, soyad, maas)
select 1000+rownum,
       'PersonelAd'||rownum,
       'PersonelSoyad'||rownum,
       round(dbms_random.value(2000, 10000),2)
from dual
connect by level <= 5;

update personel set prim = 150
where prim = 100;

-------------------------------------------
--Mutating Tables
-------------------------------------------

create or replace trigger tg_pers_mut
before insert or update of maas on personel
for each row
when (new.dept_id = 100)
declare
    v_min_maas number;
    v_max_maas number;
begin

    select min(maas), max(maas)
    into v_min_maas, v_max_maas
    from personel where dept_id = 100;

    if not (:new.maas between v_min_maas and v_max_maas) then
       raise_application_error(-20005, 'Maaş değeri sınırların dışındadır');
    end if;

end;

-------------------------------------------
--Compound Trigger İle Mutating Sorununu Çözme
-------------------------------------------

create or replace trigger tg_pers_mut
for insert or update of maas on personel
when (new.dept_id = 100)
compound trigger
    v_min_maas number;
    v_max_maas number;

    before statement is
    begin
        select min(maas), max(maas)
        into v_min_maas, v_max_maas
        from personel where dept_id = 100;
    end before statement;

    after each row is
    begin
        if not (:new.maas between v_min_maas and v_max_maas) then
           raise_application_error(-20005, 'Maaş değeri sınırların dışındadır');
        end if;
    end after each row;
end;
```

```
-------------------------------------------
--Compound Trigger - Alıştırma Scripti
-------------------------------------------

create or replace trigger trg_total_discount
    after insert or update
    on product_segment
    for each row
declare
    v_total_discount pls_integer;

begin

    -- get the total discount
    select sum (discount) into v_total_discount
    from product_segment;

    -- check total discount
    if v_total_discount + :new.discount - :old.discount > 0.25 then

        --:new.discount := :old.discount;
        update product_segment set discount = :old.discount
        where id = :new.id;

    end if;

end;
/

-------------------------------------------
--DDL Trigger - Schema
-------------------------------------------

create or replace trigger trg_no_drop
before drop on egitim.schema
begin
    raise_application_error(-20005,
    'Bu şema üzerinde herhangi bir obje drop edemezsiniz');
end;

-------------------------------------------

CREATE OR REPLACE trigger EGITIM.TRG_DDL_SCHEMA
before create or alter or drop on egitim.schema
--when (ora_dict_obj_type = 'table')
begin

    dbms_output.put_line('Event Type:'   || ora_sysevent);
    dbms_output.put_line('Object Owner:' || ora_dict_obj_owner);
    dbms_output.put_line('Object Type:'  || ora_dict_obj_type);
    dbms_output.put_line('Object Name:'  || ora_dict_obj_name);

end;
```

```
-------------------------------------------

create or replace trigger trg_ddl_schema2
before create or alter
on schema
declare
    v_sqltext  dbms_standard.ora_name_list_t;
    v_sqlcount pls_integer;

begin

    v_sqlcount := ora_sql_txt(v_sqltext);

    for i in 1..v_sqlcount loop

        dbms_output.put_line('sqltext(' || i || ')=' || v_sqltext(i));

    end loop;
end;

-------------------------------------------
--DDL Trigger – Schema – Alıştırma Scripti
-------------------------------------------

create table ddl_log
(
     operation_type varchar2(50),
     object_owner varchar2(50),
     object_name varchar2(50),
     user_name varchar2(50),
     ddl_time date,
     sql_text clob
);

-------------------------------------------
--DDL Trigger – Database
-------------------------------------------

create table log_triggering
(
    kullanci_id varchar2(30),
    log_zamani timestamp,
    islem varchar2(50)
)

create or replace trigger trg_logon
after logon
on database
begin

    insert into log_triggering
    values(USER, SYSDATE,
           'Kullanıcı Login');

end;

create or replace trigger trg_logoff
before logoff
on database
begin
    insert into log_triggering
    values(USER, SYSDATE,
           'Kullanıcı Logoff');
end;
```

```
-----------------------------------------

CREATE TABLE log_info_session
(
    username    VARCHAR2(30),
    logon_date  DATE,
    session_id  VARCHAR2(30),
    ip_addr     VARCHAR2(30),
    hostname    VARCHAR2(30),
    auth_type   VARCHAR2(30)
);

CREATE OR REPLACE TRIGGER TRG_LOGOFF
BEFORE LOGOFF ON DATABASE
DECLARE
 session_id VARCHAR2(30);
 ip_addr    VARCHAR2(30);
 hostname   VARCHAR2(30);
 auth_type  VARCHAR2(30);
BEGIN

    SELECT sys_context ('USERENV', 'SESSIONID'),
           sys_context ('USERENV', 'IP_ADDRESS'),
           sys_context ('USERENV', 'HOST'),
           sys_context ('USERENV', 'AUTHENTICATION_TYPE')
    INTO session_id, ip_addr, hostname, auth_type
    FROM dual;

    INSERT INTO log_info_session VALUES
    (user, sysdate, session_id, ip_addr, hostname, auth_type);

END;
```

```
-----------------------------------------

CREATE TABLE LOG_ALL_DDL_DB
(
  USERNAME      VARCHAR2(30),
  LOG_DATE      DATE,
  SESSION_ID    VARCHAR2(30),
  IP_ADDR       VARCHAR2(30),
  HOSTNAME      VARCHAR2(30),
  AUTH_TYPE     VARCHAR2(30),
  EVENT_TYPE    VARCHAR2(30),
  OBJECT_OWNER  VARCHAR2(30),
  OBJECT_TYPE   VARCHAR2(30),
  OBJECT_NAME   VARCHAR2(30),
  SQL_TEXT      CLOB
);

CREATE OR REPLACE TRIGGER TRG_ALL_DDL_DB
AFTER DDL ON DATABASE
declare
    session_id varchar2(30);
    ip_addr    varchar2(30);
    hostname   varchar2(30);
    auth_type  varchar2(30);
    v_sqltext  dbms_standard.ora_name_list_t;
    v_sqlcount pls_integer;
    v_ddl_text clob;
begin
    v_sqlcount := ora_sql_txt(v_sqltext);
    for i in 1..v_sqlcount loop
        v_ddl_text := v_ddl_text || v_sqltext(i);
    end loop;

    select sys_context ('USERENV', 'SESSIONID'),
           sys_context ('USERENV', 'IP_ADDRESS'),
           sys_context ('USERENV', 'HOST'),
           sys_context ('USERENV', 'AUTHENTICATION_TYPE')
    into session_id, ip_addr, hostname, auth_type
    from dual;

    insert into log_all_ddl_db values
    (user, sysdate, session_id, ip_addr, hostname, auth_type,
     ora_sysevent, ora_dict_obj_owner, ora_dict_obj_type,
     ora_dict_obj_name, v_ddl_text);
end;
```

```
-----------------------------------------

CREATE OR REPLACE TRIGGER TRG_DDL_TABLO_KONTROL
AFTER DDL ON DATABASE
declare
    tabloKontrol integer := 0;
begin

    if ORA_SYSEVENT in ('CREATE') and ORA_DICT_OBJ_TYPE = 'TABLE' then

        select count(*) into tabloKontrol
        from dba_tables
        where table_name = ORA_DICT_OBJ_NAME;

        if tabloKontrol > 1 then

            raise_application_error(-20001,
                '['||ORA_DICT_OBJ_NAME||
                '] isimli tablodan başka bir şemada da bulunmaktadır.' ||
                'Lütfen tablo ismini değiştiriniz.');

        end if;
    end if;
end;
```

```
-------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------

-----------------------------------------
--Compound Trigger
-----------------------------------------

create or replace trigger trg_total_discount
    for insert or update on product_segment
    compound trigger

    v_total_discount number;

    before statement is
    begin

        -- get the total discount
        select sum (discount) into v_total_discount
        from product_segment;

    end before statement;

    before each row is
    begin

        -- check total discount
        if v_total_discount + :new.discount - :old.discount > 0.25 then
            :new.discount := :old.discount;
        end if;

    end before each row;

end;
/

-----------------------------------------
--DDL Trigger - Schema
-----------------------------------------

create or replace trigger trg_schema_ddl
    after ddl on schema
declare
    v_sqltext_col  dbms_standard.ora_name_list_t;
    v_sqlcount pls_integer;
    v_sqltext clob;

begin

    v_sqlcount := ora_sql_txt(v_sqltext_col);
    v_sqltext := '';

    for i in 1..v_sqlcount loop

        v_sqltext := v_sqltext || v_sqltext_col(i);

    end loop;

    insert into ddl_log
    select ora_sysevent, ora_dict_obj_owner,
        ora_dict_obj_name, user, sysdate, v_sqltext
    from dual;

end;
```

```
-------------------------------------------
--DDL Trigger - Database
-------------------------------------------

create or replace trigger trg_ddl_truncate_control
before ddl on database
begin

    if ora_sysevent in ('TRUNCATE') and USER = 'TEST' and ora_dict_obj_owner = 'EGITIM' then

        raise_application_error(-20001,
            'The TEST user cannot truncate the table in the EGITIM schema!');

    end if;

end;
```

```
-------------------------------------------
--Standart: Exceptions Declaring
-------------------------------------------

create or replace package pck_excep is

    e_personel_yok exception;
    e_maas_yuksek  exception;
    e_sayi_degil   exception;

    pragma exception_init(e_personel_yok, -20001);
    pragma exception_init(e_maas_yuksek, -20002);
    pragma exception_init(e_sayi_degil, -20003);

end pck_excep;

create or replace procedure personel_bul (p_personel_id number)
is
    v_cnt number;

begin
    select count(*) into v_cnt from personel
    where personel_id = p_personel_id;

    if v_cnt = 0 then
       raise pck_excep.e_personel_yok;
    end if;
end;

begin
    personel_bul(6666);
    exception
        when pck_excep.e_personel_yok then
            raise_application_error(SQLCODE,
            'Personel bulunamadı, lütfen kontrol ediniz');
end;

-------------------------------------------
--Standart: Constants
-------------------------------------------

create or replace package pck_const is

    c_min_maas          constant number := 3500;
    c_max_maas          constant number := 15000;
    c_zam_orani         constant number := 12;
    c_ilkyil_izin_gunu  constant integer := 14;
    c_pi                constant real := 3.14159;
    c_default_bolge     constant varchar2(10) := 'MARMARA';

end pck_const;

-------------------------------------------

declare
    v_cnt number;
Begin

    select count(*) into v_cnt from personel
    where maas < pck_const.c_min_maas;

    dbms_output.put_line(v_cnt);
    dbms_output.put_line(pck_const.c_min_maas);
end;
```

```
-------------------------------------------
--Lokal Alt Programlar
-------------------------------------------

declare

    function kareal(sayi number)
    return number as
    begin
        return sayi * sayi;
    end;

begin

    dbms_output.put_line(kareal(25));

end;

-------------------------------------------

declare
    cursor crs_personel is
        select ad, soyad from personel
        where unvan = 'GRUP MÜDÜRÜ';
    v_isim varchar2(50);

    function isim_format(p_ad varchar2, p_soyad varchar2)
    return varchar2 as
    begin
        return 'Adı: ' || p_ad || ', Soyadı: ' || p_soyad;
    end;

begin

    for rowx in crs_personel
    loop
        v_isim := isim_format(rowx.ad, rowx.soyad);
        dbms_output.put_line(v_isim);
    end loop;
end;

-------------------------------------------
--Invoker Yetkileri: AUTHID CURRENT_USER
-------------------------------------------

create or replace procedure dept_ekle
(
    p_id number,
    p_isim varchar2
)
AUTHID CURRENT_USER
is
Begin

    insert into departman values(p_id, p_isim);

end;

--Connect as another user
Create departman table (with same name)
exec egitim.dept_ekle(444, 'Invoker Deneme');
```

```
-------------------------------------------
--Autonomous Transactions (AT)
-------------------------------------------

--log_iller tablosunu oluşturuyoruz

create table log_iller as
select il_kodu, il_adi, sysdate tarih from iller
where 1=0;


--Oluşturulan tabloya log yazan prosedürü yazıyoruz

create or replace procedure log_iller_yaz
    (p_ilkodu number, p_iladi varchar2)
is
    pragma autonomous_transaction;
begin
    insert into log_iller values(p_ilkodu, p_iladi, sysdate);
    commit;
end;

declare
    v_ilkodu pls_integer := 100;
    v_iladi  varchar2(30) := 'Selçuklu';
    v_bolgekodu pls_integer := 3;

begin
    insert into iller values(v_ilkodu, v_iladi, v_bolgekodu);
    log_iller_yaz(v_ilkodu, v_iladi);
    rollback;
end;

-------------------------------------------
--Performance: NOCOPY Hint
-------------------------------------------

CREATE OR REPLACE procedure personel_bilgi2
    ( p_personel_id   in  number,
      p_ad            out nocopy varchar2 )
is
begin

    select ad into p_ad from personel
    where personel_id = p_personel_id;

end;


-------------------------------------------
--Performance: PARALLEL ENABLE
-------------------------------------------

create or replace function per_isim_getir(p_id number)
return varchar2 parallel_enable
is
    v_isim varchar2(40);
begin
    select ad into v_isim
    from personel
    where personel_id = p_id;

    return v_isim;
end;
```

```
-------------------------------------------
--Performance: RESULT_CACHE
-------------------------------------------

create or replace function per_isim_getir(p_id number)
return varchar2 result_cache
is
    v_isim varchar2(40);
begin
    select ad into v_isim
    from personel
    where personel_id = p_id;

    return v_isim;
end;


-------------------------------------------
--Performance: DETERMINISTIC
-------------------------------------------

create or replace function pass_number (i number)
   return number
   deterministic
is
begin
   dbms_output.put_line ('pass_number çalıştırıldı: '||i);
   return i;
end;

-------------------------------------------

declare
   n   number := 0;
begin
   for rec in (select pass_number (1) from dual
               connect by level < 5)
   loop
      n := n + 1;
   end loop;

   dbms_output.put_line (n);
end;


-------------------------------------------
--Performance: RETURNING
-------------------------------------------

create or replace procedure maas_guncelle (p_id number)
is
    v_sonuc varchar2(50);
begin

    update personel
    set maas = maas * 1.15
    where personel_id = p_id
    returning ad||' '||soyad || ' : '||maas into v_sonuc;

    dbms_output.put_line(v_sonuc);
end;
```

```
-------------------------------------------

declare
    v_sonuc varchar2(50);
begin

    insert into konum
    values(sq_temp.nextval, 'Erzurum Tortum', 25)
    returning konum_id||', '||konum_adi || ', '||il_kodu into v_sonuc;

    dbms_output.put_line(v_sonuc);
end;


-------------------------------------------
--Performance: Bulk Binding
-------------------------------------------

create table bulk_bind_table1 (sayi number, deger varchar2(20));
create table bulk_bind_table2 (sayi number, deger varchar2(20));

create or replace procedure bind_table_ornek is
    type type_sayi  is table of number       index by pls_integer;
    type type_deger is table of varchar2(20) index by pls_integer;
    v_sayi type_sayi;
    v_deger type_deger;

    sayac constant pls_integer := 100000;
    t1 integer;
    t2 integer;
    t3 integer;
begin
    for i in 1..sayac loop
        v_sayi(i) := i;
        v_deger(i) := 'Değer No: '||i;
    end loop;
    t1 := dbms_utility.get_time;
    for j in 1..sayac loop
        insert into bulk_bind_table1 values(v_sayi(j), v_deger(j));
    end loop;
    t2 := dbms_utility.get_time;
    forall j in 1..sayac
        insert into bulk_bind_table2 values(v_sayi(j), v_deger(j));
    t3 := dbms_utility.get_time;

    dbms_output.put_line('For Loop: ' || to_char((t2-t1)/100));
    dbms_output.put_line('ForAll : '  || to_char((t3-t2)/100));
    commit;
end;
```

```
-------------------------------------------
--Performance: Sorgu İçinde Bulk Collect Into
-------------------------------------------

create or replace procedure konum_getir
is
    type konum_type is table of konum%rowtype;

    v_konum konum_type;

begin

    select * bulk collect into v_konum from konum order by 1;

    for i in 1..v_konum.count loop
        dbms_output.put_line(v_konum(i).konum_id || ' ' ||
            v_konum(i).konum_adi);
    end loop;

end;

-------------------------------------------
--Performance: Returning İle Bulk Collect Into
-------------------------------------------

create or replace procedure maaslara_zam_yap(zam_orani number)
is
    type perid_type is table of number;
    type maas_type is table of personel.maas%type
        index by pls_integer;

    v_perid perid_type;
    v_yeni_maaslar maas_type;

begin

    select personel_id bulk collect into v_perid
    from personel order by personel_id;

    forall i in v_perid.first..v_perid.last
        update personel set maas = maas * zam_orani
        where personel_id = v_perid(i)
        returning maas bulk collect into v_yeni_maaslar;

    for i in 1..v_yeni_maaslar.count loop
        dbms_output.put_line(v_perid(i) || ': ' || round(v_yeni_maaslar(i),2));
    end loop;

end;
```

```
-------------------------------------------------
--ALIŞTIRMALARIN CEVAPLARI
-------------------------------------------------

-----------------------------------------
--Lokal Alt Programlar
-----------------------------------------

create or replace function create_emp_email (emp_id number)
return varchar2
is
    v_first_name employees.first_name%type;
    v_last_name  employees.last_name%type;
    v_email      varchar2(27);
    v_hosting    varchar2(12) := 'dbhunter.net';
    v_return_msg varchar2(200);

    -- declare and define procedure

    procedure create_email
    (
        name1   varchar2,
        name2   varchar2,
        company varchar2
    )
    is
        error_message varchar2(35) := '...email address is too long...';

    begin

        v_email := lower(name1) || '.' || lower(name2) || '@' || company;
        v_return_msg := v_email;

        exception
        when value_error then
          v_return_msg := error_message;

    end create_email;

begin

    select first_name, last_name
        into v_first_name, v_last_name
    from employees
    where employee_id = emp_id;

    create_email(v_first_name, v_last_name, v_hosting);

    return v_return_msg;

end;
/
```