Adatbázisrendszerek II. – 8. Practice

Topic: Oracle PL/SQL programozás, PLSQL 1

Repository: NEPTUNKOD_DB2Gyak

Folder: NEPTUNKOD_0403

Forrás file-k: forrás fájlok **.SQL**

Töltse fel a GitHub rendszer aktuális mappába a jegyzőkönyvet és forrás fájlokat!

Határidő: aktuális gyakorlat időpontja, ill. módosítás esetén 2025.04.01.

Feladatok

Projekt név: NEPTUNKOD_PLSQL1

Csomagnév: neptunkoddb2

Feladat

1. Készítsen egy ErtekNovel(sz) nevű tárolt eljárást a **Zoo** táblához, mely egy paraméterben megadott százalékos értékkel megnöveli minden állat értékét!
2. Készítsen egy ErtekCsokkent(sz) nevű tárolt eljárást, mely egy paraméterben megadott százalékos értékkel megnövelt értéket minden állatnál visszaállít az eredeti értékre!
3. Készítsen egy OsszSuly(g) nevű függvényt, mely egy paraméterben megadott nevű gondozó állatainak összsúlyát adja eredményül!
4. Készítsen egy OsszErtek(g) nevű függvényt, mely egy gondozó állatainak összértékét adja eredményül!
5. Készítsen egy Darab(g) nevű függvényt, mely egy gondozó állatainak darabszámát adja eredményül!
6. Készítsen egy Statisztika(g) nevű eljárást, mely egy gondozó statisztikai adatait kiírja a képernyőre.
   - Az adatok kiszámításához használja az előző függvényeket!
   - Kiírási forma:

```
Gondozó: Kis János
Állatok száma: 5
Állatok összsúlya: 8420
Állatok összértéke: 13700000
```

Minden alprogramot néhány adaton próbáljon ki!

Javaslat a feladatok elkészítéséhez TXT fájl.

Minta a csatolt fájl elkészítésére:

```
create or replace proc ErtekNovel(x in int) as
 . . . //az eljárás forráskódja
}
================================ // a vonal is kell!!
begin
  ErtekNovel(10);
end;   //a futtatás kódja(i)
================================
create or replace proc ErtekCsokkent(x in int) as
 . . . //az eljárás forráskódja
}
================================
begin
  ErtekCsokkent(10);
end;   //a futtatás kódja(i)
================================
   . . .
```

Körök adatai

Feladat: készíteni kell egy táblát, és feltölteni **x** és **y** közötti sugarú körök adataival: *sugár, kerület és terület.*

Úgy tűnik, ezt SQL paranccsal nem lehet megoldani, sok-sok SQL paranccsal talán igen, de mégsem, bár lehet…

Circles		
Radius	Circumference	Area

A táblát létrehozó parancs:

```
CREATE TABLE Circles (
  Radius number(4) PRIMARY KEY,
  Circumference number,
  Area number
);
```

Próbáljuk egy kör kerületét meghatározni:

– Vajon van Oracleben PI függvény: Nincs! Ezért:

```
SELECT 1 Radius, 2*1*3.141592654 Circumference from dual;
```

RADIUS	CIRCUMFERENCE
1	6.283185308

Sokat fogunk PI-zni, ezért írjunk egy PI() függvényt!

```
CREATE OR REPLACE Function PI RETURN number AS
BEGIN
  RETURN 3.141592654;
END;
```

```
Function created.
```

Próbáljuk egy kör kerületét meghatározni:

– Már van `PI()` függvényünk, ezért:

```
SELECT 1 Radius, 2*1*PI() Circumference from dual;
```

RADIUS	CIRCUMFERENCE
1	6.283185308

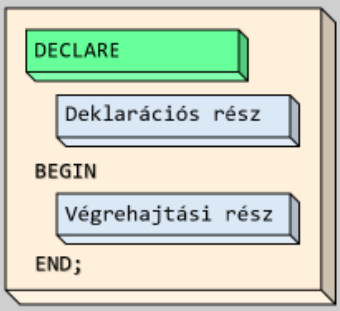
Most próbáljuk elhagyni az SQL-t, és írjuk meg ugyanezt PLSQL nyelven:

```
DECLARE
  Radius number:= 1;
  Circumference number;
BEGIN
  Circumference:= 2*Radius*PI();
  DBMS_OUTPUT.PUT_LINE('Radius: '||Radius||' Circumference: '
                                                ||Circumference);
END;
```

```
Radius: 1 Circumference: 6.283185308
```

PLSQL BLOKK - magyarázat

```
DECLARE
  Radius number:= 1;
  Circumference number;
BEGIN
  Circumference:= 2*Radius*PI();
  DBMS_OUTPUT.PUT_LINE('Radius: '||Radius||' Circumference: '
                                                ||Circumference);
END;
```



Ez egy egyszer használható PLSQL BLOKK.

A változókat a DECLARE és a BEGIN között kell deklarálni.

Ha nincs deklaráció, a blokk a BEGINnel kezdődik.

`DBMS_OUTPUT.PUT_LINE('xx')` – Kiírás a képernyőre

`||` – Összefűzés a kiíró utasításban

Körök adatai

Próbáljuk több kör kerületét meghatározni (kell egy ciklus!):

```
DECLARE
  Circumference number;
  x number:= 1;
  y number:= 5;
BEGIN
  FOR i IN x..y LOOP
    Circumference:= 2*i*PI();
    DBMS_OUTPUT.PUT_LINE('Radius: '||i||' Circumference: '
                                                  ||Circumference);
  END LOOP;
END;
```

```
Radius: 1 Circumference: 6.283185308
Radius: 2 Circumference: 12.566370616
Radius: 3 Circumference: 18.849555924
Radius: 4 Circumference: 25.132741232
Radius: 5 Circumference: 31.41592654
```

Körök adatai – tárolt eljárás

Hogy ne kelljen mindig megírni a blokkot, írjunk belőle egy tárolt eljárást:

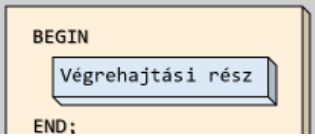
```
CREATE OR REPLACE Procedure Circler (x in number, y in number) IS
  Circumference number;
BEGIN
  FOR i IN x..y LOOP
    Circumference:= 2*i*PI();
    DBMS_OUTPUT.PUT_LINE('Radius: '||i||' Circumference: '
                                                  ||Circumference);
  END LOOP;
END;
```

```
Procedure created.
```

A szerveren létrejött, és tárolódott az eljárás.

Indítsuk el.

```
BEGIN
  Circler(1, 5);
END;
```



Ez egy tárolt eljárást indító PLSQL BLOKK.

Körök adatai – terület kiírás

Jó lenne a területet is kiírni:

```
CREATE OR REPLACE Procedure Circler (x in number, y in number) IS
  Circumference number;
  Area number;
BEGIN
  FOR i IN x..y LOOP
    Circumference:= 2*i*PI();
    Area:= POWER(i, 2)*PI();
    DBMS_OUTPUT.PUT_LINE('Radius: '||i||' Circumference: '
                                  ||Circumference||' Area: '||Area);
  END LOOP;
END;
```

```
BEGIN
  Circler(1, 5);
END;
```

```
Radius: 1 Circumference: 6.283185308 Area: 3.141592654
Radius: 2 Circumference: 12.566370616 Area: 12.566370616
Radius: 3 Circumference: 18.849555924 Area: 28.274333886
Radius: 4 Circumference: 25.132741232 Area: 50.265482464
Radius: 5 Circumference: 31.41592654 Area: 78.53981635
```

Körök adatai – adatok a táblába

Már csak bele kell írni az adatokat a táblába:

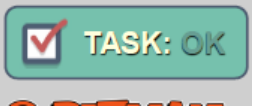
```
CREATE OR REPLACE Procedure Circler (x in number, y in number) IS
  Circumference number;
  Area number;
BEGIN
  FOR i IN x..y LOOP
    Circumference:= 2*i*PI();
    Area:= POWER(i, 2)*PI();
    INSERT INTO Circles VALUES(i, Circumference, Area);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('A Circles táblába bekerültek az adatok.');
END;
```

```
BEGIN
  Circler(1, 5);
END;
```

```
A Circles táblába bekerültek az adatok.
```

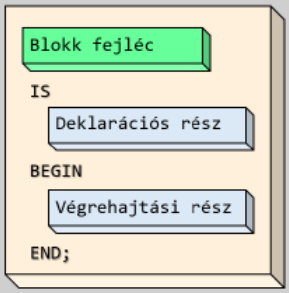
```
SELECT * FROM CIRCLES
```

RADIUS	CIRCUMFERENCE	AREA
1	6.283185308	3.141592654
2	12.566370616	12.566370616
3	18.849555924	28.274333886
4	25.132741232	50.265482464



PLSQL nevesített, tárolt BLOKK

```
CREATE OR REPLACE Procedure Circler (x in number, y in number) AS
  Circumference number;
BEGIN
  FOR i IN x..y LOOP
    Circumference:= 2*i*PI();
    DBMS_OUTPUT.PUT_LINE('Radius: '||i||' Circumference: '||Circumference);
  END LOOP;
END;
```



Ezek a tárolt eljárások, és függvények

Megírjuk, elindítjuk, a rendszer lefordítja futtatható kódra, és letárolja a szerveren a forrás- és a futtatható kódot is

Az Auto tábla kódja

Auto						
ID	Rendszam	Tipus	Szin	Kor	Ar	Muszaki_erv

Az itt bemutatott feladatok az `Auto` tábla adatai alapján lettek megoldva.

Javaslom: hozza létre a táblát, hozza létre, és futtassa le a bemutatott tárolt eljárásokat.

Ezután hozza létre a Zoo táblát, töltse fel adatokkal, és próbáljon a bemutatotthoz hasonló tárolt eljárást készíteni a Zoo táblához!

Az `Auto` tábla létrehozási parancsa:

```
CREATE TABLE Auto (
   ID number(4) PRIMARY KEY,
   Rendszam char(6),
   Tipus varchar2(30),
   Szin varchar2(20),
   Kor number(3),
   Ar number(10),
   Muszaki_erv date
);
```

Az Auto tábla feltöltése

Auto						
ID	Rendszam	Tipus	Szin	Kor	Ar	Muszaki_erv

■ A minta tábla feltöltése:

```
BEGIN
  INSERT INTO Auto VALUES(1,'ABC123','Opel Corsa','Piros',6,3200000,'05.12.2024');
  INSERT INTO Auto VALUES(2,'QWJ612','Suzuki Alto','Kék',15,500000,'03.28.2024');
  INSERT INTO Auto VALUES(3,'PHD229','Skoda Scala','Fehér',1,6800000,'10.06.2025');
  INSERT INTO Auto VALUES(4,'PET220','Opel Astra','Fekete',3,4400000,'08.05.2024');
  INSERT INTO Auto VALUES(5,'RWE611','Ford Focus','Sárga',4,5300000,'01.15.2025');
  INSERT INTO Auto VALUES(6,'MNS418','Skoda Karoq','Kék',2,7000000,'02.28.2025');
  INSERT INTO Auto VALUES(7,'MUX312','Suzuki Swift','Piros',8,1600000,'10.23.2024');
  INSERT INTO Auto VALUES(8,'NCO927','Tesla Y','Fehér',3,26000000,'05.26.2024');
END;
```

Írassa ki a szerveren a rendszerdátumot. Látszik: a dátum mm/dd/yyyy alakú. Ilyen sorrendben kell felvitelkor megadni a dátum egyes elemeit.

```
SELECT sysdate FROM dual;
```

SYSDATE
04/11/2023

A Zoo tábla kódja

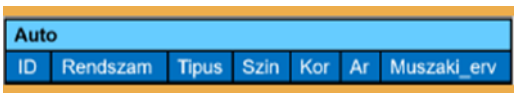

```
CREATE TABLE Zoo (
  ID int primary key,
  Nev varchar2(30),
  Fajta varchar2(30),
  Szuletett date,
  Suly int,
  Ertek int,
  Gondozo varchar2(30)
);
```

A Zoo tábla feltöltése


```
BEGIN
INSERT INTO Zoo Values(1, 'Nelly', 'Elefánt', '05.09.1999', 3500, 4000000, 'Kis János');
INSERT INTO Zoo Values(2, 'Molly', 'Majom', '04.24.2014', 40, 300000, 'Kis János');
INSERT INTO Zoo Values(3, 'Decker', 'Elefánt', '02.07.2012', 2200, 3000000, 'Kis János');
INSERT INTO Zoo Values(4, 'Dotty', 'Zsiráf', '05.09.1999', 890, 3000000, 'Kis János');
INSERT INTO Zoo Values(5, 'Betsy', 'Teve', '03.23.2004', 1750, 4000000, 'Kis János');
INSERT INTO Zoo Values(6, 'Lee', 'Tigris', '01.19.2017', 350, 800000, 'Kemény Péter');
INSERT INTO Zoo Values(7, 'Hank', 'Zsiráf', '05.17.1992', 930, 2400000, 'Kis János');
INSERT INTO Zoo Values(8, 'Foxi', 'Róka', '12.05.2020', 55, 250000, 'Kemény Péter');
INSERT INTO Zoo Values(9, 'Neo', 'Pingvin', '05.18.2018', 40, 100000, 'Boldog Éva');
INSERT INTO Zoo Values(10, 'Bruce', 'Cápa', '07.26.2006', 280, 1000000, 'Boldog Éva');
INSERT INTO Zoo Values(11, 'Olivia', 'Krokodil', '10.20.2006', 450, 100000, 'Boldog Éva');
INSERT INTO Zoo Values(12, 'Coca', 'Koala', '07.01.2016', 200, 10000000, 'Boldog Éva');
INSERT INTO Zoo Values(13, 'Bruce', 'Majom', '11.29.2011', 40, 1000000, 'Boldog Éva');
INSERT INTO Zoo Values(14, 'Benny', 'Medve', '10.05.2020', 80, 1500000, 'Kemény Péter');
INSERT INTO Zoo Values(15, 'Brumi', 'Plüssmaci', '10.02.2016', 1, 12000, 'Ági baba');
END;
```

Tárolt eljárások

Átlag()



Írjunk egy átlagárat kiíró PLSQL eljárást!

```
CREATE OR REPLACE Procedure Atlag AS
  x Auto.Ar%TYPE;
BEGIN
  SELECT AVG(Ar) INTO x FROM Auto;
  DBMS_OUTPUT.PUT_LINE(x);
END;
```

```
BEGIN
  Atlag;
END;
```

```
6857778
```

PLSQL – Mi micsoda?

```
CREATE OR REPLACE Procedure Atlag AS
  x Auto.Ar%TYPE;
BEGIN
  SELECT AVG(Ar) INTO x FROM Auto;
  DBMS_OUTPUT.PUT_LINE(x);
END;
```

Kötelező kulcsszavak
Eljárás neve
Változó deklaráció
Kiíratás képernyőre
SQL parancs
PLSQL utasítás

PLSQL – Mi micsoda?

```
CREATE OR REPLACE Procedure Atlag AS
  x Auto.Ar%TYPE;
BEGIN
  SELECT AVG(Ar) INTO x FROM Auto;
  DBMS_OUTPUT.PUT_LINE(x);
END;
```

CREATE OR REPLACE: hozd létre, vagy ha már létezik írd felül
A CREATE önállóan is használható (OR REPLACE nélkül)

A tárolt eljárás típusa lehet:
- Procedure: eljárás (nincs visszatérő értéke, vagy több is van)
- Function: függvény (van visszatérő értéke)

PLSQL – Mi micsoda?

```
CREATE OR REPLACE Procedure Atlag AS
  x Auto.Ar%TYPE;
BEGIN
  SELECT AVG(Ar) INTO x FROM Auto;
  DBMS_OUTPUT.PUT_LINE(x);
END;
```

Atlag: az eljárás neve. Utaljon a név a tartalomra.

AS: ezután következik az eljárás kódja. Használható az AS helyett az IS kulcsszó is.

x Auto.Ar%TYPE; Változó deklaráció típusmegadással
- x – a változó neve
- Auto.Ar%TYPE – a változó típusa másolással jön létre: ugyanolyan lesz a típus, mint az Auto tábla Ar mezőjének a típusa

Mindig, minden változót deklarálni kell! A változók deklarációja mindig a BEGIN és az eljárás deklarációs sora közé kerül.

PLSQL – Mi micsoda?

```
CREATE OR REPLACE Procedure Atlag AS
  x Auto.Ar%TYPE;
BEGIN
  SELECT AVG(Ar) INTO x FROM Auto;
  DBMS_OUTPUT.PUT_LINE(x);
END;
```

A lekérdezett átlagárat berakjuk az x változóba (INTO x).

DBMS_OUTPUT.PUT_LINE(): kiírja a képernyőre az adatot. Csak változók értékét, és konstans szövegeket lehet kiírni, műveletek eredményét közvetlenül nem. Ezért **hibás** a következő utasítás:
DBMS_OUTPUT.PUT_LINE(SELECT AVG(Ar) FROM Auto);

Átlag() - a kód finomítása

Auto						
ID	Rendszam	Tipus	Szin	Kor	Ar	Muszaki_erv

Pici módosítás, hogy érthető legyen a kiírt adat:

```
CREATE OR REPLACE Procedure Atlag AS
  x Auto.Ar%TYPE;
  xhu varchar(30);
BEGIN
  SELECT AVG(Ar) INTO x FROM Auto;
  xhu:= TO_CHAR(x, '9G999G999G999');
  xhu:= LTRIM(REPLACE(xhu, ',', ' '));
  DBMS_OUTPUT.PUT_LINE('Az autók átlagára: '||xhu||' Ft');
END;
```

```
BEGIN
  Atlag;
END;
```

```
Az autók átlagára: 6 857 778 Ft
```

Magyarázatok

```
xhu:= TO_CHAR(x, '9G999G999G999');
```

TO_CHAR() - numerikus adat konvertálása megadott formátummal

'9G999G999G999' - Ezres csoportokat kialakító formátum

Mivel a szerveren amerikai számformátum van beállítva, a szám így 1,234,567 alakú lesz

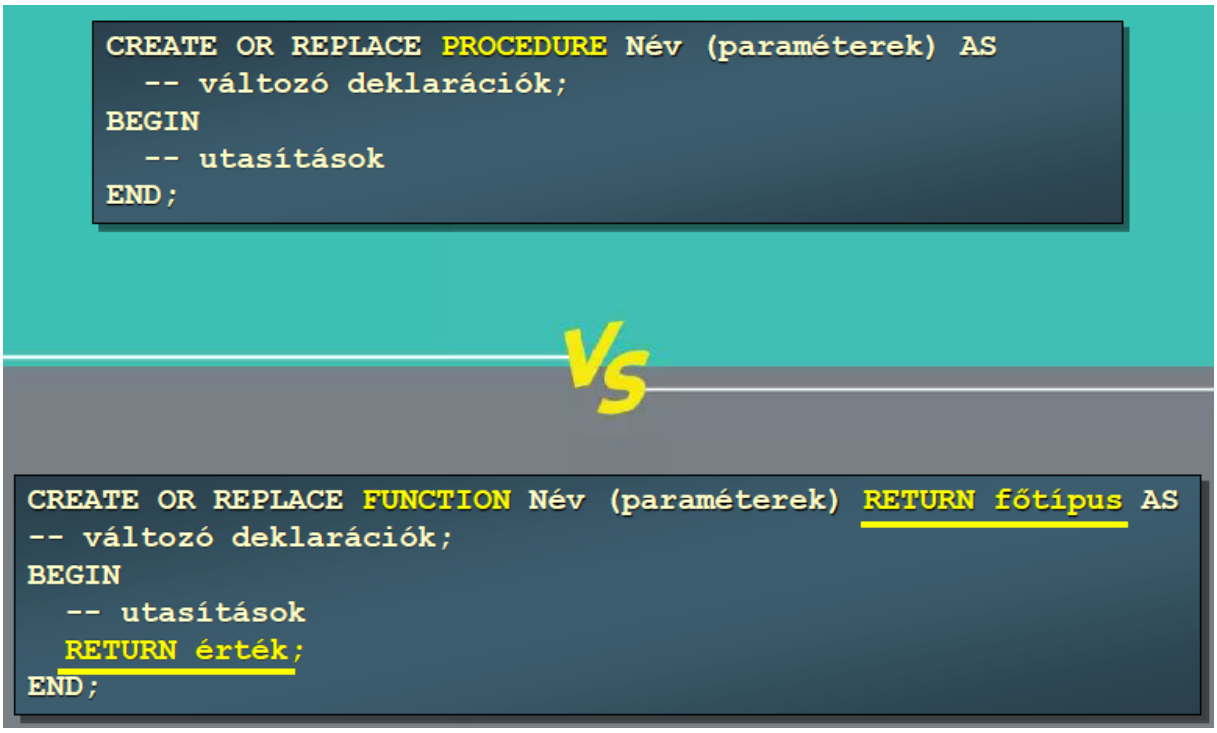
```
xhu:= LTRIM(REPLACE(xhu, ',', ' '));
```

REPLACE(s,w,o) - karakter cseréje szövegben (vesszőket szóközre)

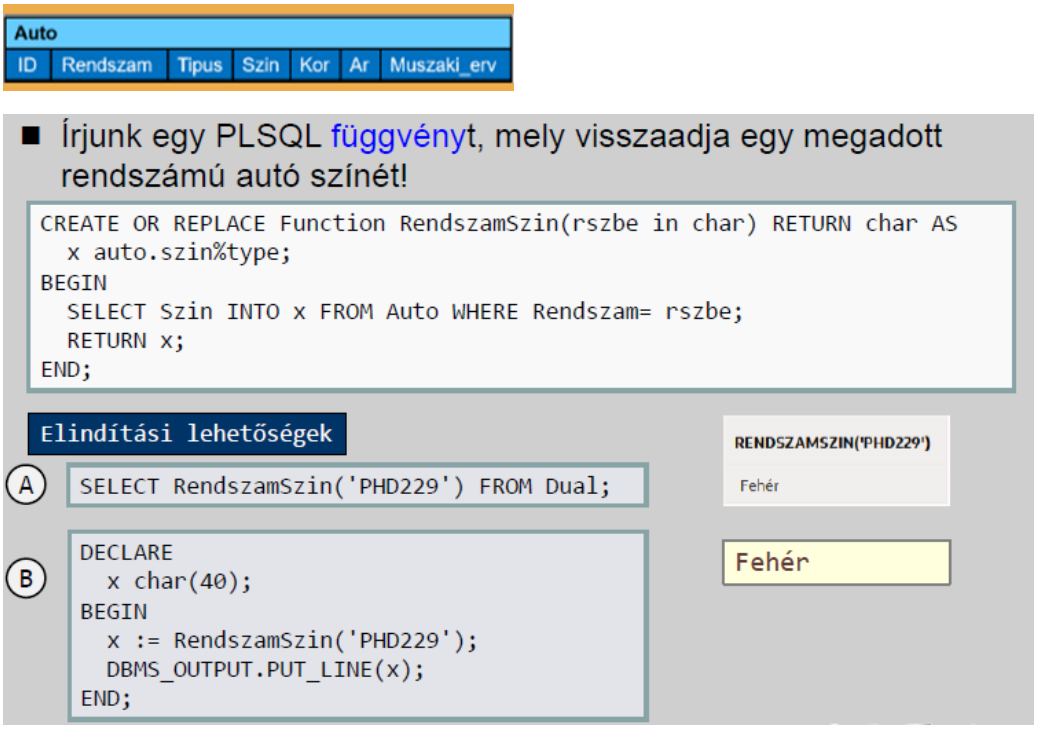
```
xhu:= LTRIM(REPLACE(xhu, ',', ' '));
```

LTRIM(s) - szóközök törlése bal oldalról

Procedure vs. Function



RendszámSzín



PLSQL – Mi micsoda?

```
CREATE OR REPLACE Function RendszamSzin (rszbe in char)
                                                RETURN char AS
  x Auto.Szin%TYPE;
BEGIN
  SELECT Szin INTO x FROM Auto WHERE Rendszam= rszbe;
  RETURN x;
END;
```

- Kötelező kulcsszavak
- Függvény neve
- Bemenő paraméter megadása
- Visszatérő érték típusának megadása
- Visszatérő érték megadása

PLSQL – Mi micsoda?

```
CREATE OR REPLACE Function RendszamSzin (rszbe in char)
                                                RETURN char AS
  x Auto.Szin%TYPE;
BEGIN
  SELECT Szin INTO x FROM Auto WHERE Rendszam= rszbe;
  RETURN x;
END;
```

A legfontosabb különbség az eljárás és a függvény között:

- az alprogram típusa (Function),
- meg kell adni a RETURN kulcsszóval a visszatérő érték típusát,
- és legutolsó kódsorként a RETURN paranccsal vissza kell adni egy értéket.

A RETURN után visszaadott érték típusának meg kell egyeznie a fejlécben megadott visszatérő típussal!

PLSQL – Mi micsoda?

```
CREATE OR REPLACE Function RendszamSzin (rszbe in char)
                                          RETURN char AS
  x Auto.Szin%TYPE;
BEGIN
  SELECT Szin INTO x FROM Auto WHERE Rendszam= rszbe;
  RETURN x;
END;
```

Az alprogramoknak lehetnek paraméterei. A paramétereket a név után zárójelben kell megadni:

- név jelleg típus adatokkal:
  - jelleg: lehet in, out, in out
    - in: bemenő paraméter, az in alapértelmezés, ezért az in kulcsszó elhagyható
    - out: visszatérő paraméter, egy adatot adhat vissza
    - in out: be- és kimenő paraméter, egy adat számára
  - típus: csak a főtípus adható meg, a méret megadása hibás, pl: a char(20) vagy a number(4) hibás!

ÁtlagÁr

Auto

ID	Rendszam	Tipus	Szin	Kor	Ar	Muszaki_erv

```
CREATE OR REPLACE Function AtlagAr RETURN int AS
   atlag int:= 0;
BEGIN
   SELECT avg(Ar) INTO atlag FROM Auto;
   RETURN atlag;
END;
```

Elindítási lehetőségek

A)
```
SELECT AtlagAr from dual;
```

ATLAG
2975000

B)
```
declare
  atl int;
begin
  atl := AtlagAr;
  dbms_output.put_line('Az átlag: '||atl);
end;
```

```
Az átlag: 2975000
Statement processed.
```