CREATE OR REPLACE FUNCTION dagen\_einde\_maand

RETURN NUMBER

AS

BEGIN

RETURN last\_day(sysdate) - sysdate;

END;

//MAANDEN Tussen

CREATE OR REPLACE FUNCTION aantal\_dienstjaren

(p\_hire\_date employees.hire\_date%type)

RETURN NUMBER

AS

BEGIN

RETURN TRUNC(months\_between(sysdate, p\_hire\_date)/12);

END;

/

//netto

CREATE OR REPLACE FUNCTION netto

(p\_sal employees.salary%type)

RETURN VARCHAR2

AS

v\_netto employees.salary%type;

BEGIN

v\_netto:=p\_sal\*0.6;

RETURN to\_char(v\_netto,'9,999.00') || 'EUR';

END;

/

CREATE OR REPLACE FUNCTION netto

(p\_sal employees.salary%type)

RETURN VARCHAR2

AS

v\_netto employees.salary%type;

BEGIN

IF p\_sal <= 10000 THEN

v\_netto:=p\_sal \* 0.6;

ELSIF p\_sal <= 16000 THEN

v\_netto:= 10000 \* 0.6 + (p\_sal-10000) \* 0.5;

ELSE

v\_netto:= 10000 \* 0.6 + 6000 \* 0.5 + (p\_sal-16000) \*0.4;

END IF;

RETURN to\_char(v\_netto,'9,999.00') || 'EUR';

END;

/

Schrikkeljaar

CREATE OR REPLACE FUNCTION schrikkeljaar

RETURN VARCHAR2

AS

V\_datum date;

BEGIN

v\_datum:=to\_date('01-02-'|| to\_char(sysdate,'yyyy'),'dd-mm-yyyy');

v\_datum:= last\_day(v\_datum);

IF to\_char(v\_datum,'dd')='29' THEN

RETURN 'het jaar' || to\_char(sysdate, 'yyyy') || 'is een schrikkeljaar';

ELSE

RETURN 'het jaar' || to\_char(sysdate, 'yyyy') || 'is geen schrikkeljaar';

END IF;

END;

/

CREATE OR REPLACE FUNCTION schrikkeljaar

(p\_jaar dual.sysdate%type)

RETURN VARCHAR2

AS

BEGIN

IF MOD(p\_jaar,4)=0 AND MOD(p\_jaar,100)!=0 AND MOD(p\_jaar, 400)=0 THEN

RETURN to\_char("het is een schrikkeljaar");

ELSE

RETURN to\_char("het is geen schrikkeljaar");

END IF;

END;

/

//zoveeste van de maand

CREATE OR REPLACE FUNCTION vergadering

RETURN DATE

AS

v\_maandag DATE;

BEGIN

v\_maandag:=last\_day(sysdate)+1;

IF to\_char(v\_maandag, 'fmday')!='monday' THEN

v\_maandag:= next\_day(v\_maandag, 'mon');

END IF;

IF to\_char(v\_maandag, 'dd-mm') = '01-01' OR to\_char(v\_maandag,'dd-mm')= '01-05' THEN

v\_maandag:=v\_maandag+1;

END IF;

RETURN v\_maandag;

END;

/

CREATE OR REPLACE FUNCTION next\_employee\_id

RETURN employees.employee\_id%type

AS

v\_number employees.employee\_id%type;

BEGIN

SELECT max(employee\_id INTO v\_number)

FROM employees;

RETURN v\_number +1;

END;

/

//bruto

CREATE OR REPLACE FUNCTION bruto\_jaarloon

(p\_employeeid employees.employee\_id%type)

RETURN NUMBER

AS

v\_jaarsal NUMBER;

BEGIN

SELECT (salary + salary\*NVL(salary\*commission\_pct,0))\*12 INTO v\_jaarsal

FROM employees

WHERE employee\_id = p\_employeeid;

RETURN v\_jaarsal;

END;

/

CREATE OR REPLACE FUNCTION hoofd\_meeste\_wn

RETURN VARCHAR2

AS

v\_dep\_id employees.department\_id%type;

v\_aantal NUMBER;

v\_manager\_id employees.manager\_id%type;

v\_first employees.first\_name%type;

v\_last employees.last\_name%type;

BEGIN

SELECT department\_id,count(employee\_id)

INTO v\_dep\_id,v\_aantal

FROM employees

GROUP BY department\_id

HAVING count(employee\_id) = (SELECT max(count(employee\_id))

FROM employees

GROUP BY department\_id);

SELECT manager\_id

INTO v\_manager\_id

FROM departments

WHERE department\_id = v\_dep\_id;

SELECT manager\_id

INTO v\_manager\_id

FROM departments

WHERE department\_id = v\_dep\_id;

SELECT first\_name, last\_name

INTO v\_first, v\_last

FROM employees

WHERE employee\_id=v\_manager\_id

RETURN v\_first || '' || v\_last;

END;

/

//Jubilieum

create or replace function get\_jubileumdate

(p\_last\_name employees.last\_name%type,

p\_first\_name employees.first\_name%type)

RETURN varchar2

as

v\_hire\_date date;

v\_jubileum date;

BEGIN

select hire\_date

into v\_hire\_date

from employees

where last\_name = p\_last\_name

and first\_name = p\_first\_name;

v\_jubileum := add\_months(v\_hire\_date , 25\*12);

if to\_char(v\_jubileum, 'd') != 6 then

v\_jubileum := next\_day(v\_jubileum, 'fri');

end if;

if v\_jubileum < sysdate then

return 'reeds gevierd';

else

return to\_char(v\_jubileum);

end if;

END;

/

CREATE OR REPLACE PROCEDURE grootste\_dept

AS

v\_dept\_id employees.department\_id%TYPE;

v\_dept\_naam departments.department\_name%TYPE;

v\_naam employees.last\_name%TYPE;

v\_vnaam employees.first\_name%TYPE;

v\_salaris employees.salary%TYPE;

BEGIN

SELECT department\_name, department\_id

INTO v\_dept\_naam, v\_dept\_id

FROM departments join employees

USING(department\_id)

GROUP BY department\_name, department\_id

HAVING COUNT(employee\_id)=(SELECT MAX(COUNT(\*))

FROM employees

GROUP BY department\_id);

SELECT last\_name, first\_name, salary

INTO v\_naam, v\_vnaam, v\_salaris

FROM employees JOIN departments using (department\_id)

WHERE department\_name=v\_dept\_naam

AND employees.salary = (SELECT max(salary)

FROM employees

WHERE department\_id=v\_dept\_id);

DBMS\_OUTPUT.PUT\_LINE('Departement:'||v\_dept\_naam||'hoogste salaris'||v\_naam||''||v\_vnaam||' '||v\_salaris);

END;

/

//kleinste waarde

CREATE OR REPLACE PROCEDURE minimumloon

( p\_land IN countries.country\_name%TYPE, p\_minloon IN employees.salary%type)

IS

BEGIN

UPDATE employees

SET salary = p\_minloon

WHERE department\_id =

(SELECT department\_id

FROM departments

WHERE location\_id=

(SELECT location\_id

FROM locations

WHERE country\_id=(SELECT country\_id

FROM countries

WHERE UPPER(country\_name)=UPPER(p\_land))))

AND salary < p\_minloon;

END;

/

CREATE OR REPLACE PROCEDURE minimumloon

( p\_land IN countries.country\_name%TYPE, p\_minloon IN employees.salary%type, p\_aantal OUT number)

IS

BEGIN

UPDATE employees

SET salary = p\_minloon

WHERE department\_id =

(SELECT department\_id

FROM departments

WHERE location\_id=

(SELECT location\_id

FROM locations

WHERE country\_id=(SELECT country\_id

FROM countries

WHERE UPPER(country\_name)=UPPER(p\_land))))

AND salary < p\_minloon;

p\_aantal := SQL%rowcount;

END;

/

CREATE OR REPLACE PROCEDURE landenlijst

IS

v\_aantal NUMBER(5);

v\_landnaam VARCHAR2(50);

v\_kleinste VARCHAR2(4);

BEGIN

SELECT COUNT(\*)

INTO v\_aantal

FROM COUNTRIES;

SELECT MIN(country\_id)

INTO v\_kleinste

FROM countries;

FOR i IN 1 .. v\_aantal LOOP

SELECT country\_name into v\_landnaam

FROM countries

WHERE country\_id = v\_kleinste;

DBMS\_OUTPUT.PUT\_LINE(v\_landnaam);

SELECT MIN(country\_id) INTO v\_kleinste

FROM countries

WHERE country\_id > v\_kleinste;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('We hebben connecties in '||v\_aantal||' landen.');

END landenlijst;

/

CREATE OR REPLACE PROCEDURE landenlijst\_cursor

IS

v\_aantal\_landen integer(4) := 0;

BEGIN

FOR lijn IN (SELECT country\_name

FROM countries)

LOOP

DBMS\_OUTPUT.PUT\_LINE(lijn.country\_name);

v\_aantal\_landen := v\_aantal\_landen + 1;

END LOOP;

/\* DBMS\_OUTPUT.PUT\_LINE('We hebben connecties in '|| lijn%rowcount ||' landen.');

werkt niet omdat de query geen cursor is die expliciet gedeclareerd kan worden(we kunnen er geen naam aan geven),

daarom een counter gebruiken! \*/

DBMS\_OUTPUT.PUT\_LINE('We hebben connecties in '|| v\_aantal\_landen ||' landen.');

END;

/

CREATE OR REPLACE PROCEDURE country\_dept\_cursorloop

(p\_landid IN countries.country\_id%type)

AS

v\_max number(5);

v\_laagste\_id departments.department\_id%type;

v\_dept\_id departments.department\_id%type;

v\_counter number(3) :=1;

v\_dept\_name departments.department\_name%type;

BEGIN

SELECT COUNT(location\_id) INTO v\_max

FROM departments join locations

USING (location\_id)

WHERE country\_id = p\_landid;

IF v\_max = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Er zijn geen departementen gevestigd in het land met id '||p\_landid);

ELSE

FOR rec IN (SELECT department\_name

FROM departments d join locations l

USING (location\_id)

WHERE country\_id = p\_landid)

LOOP

DBMS\_OUTPUT.PUT\_LINE(rec.department\_name);

END LOOP;

END IF;

END country\_dept\_cursorloop;

/

CREATE OR REPLACE PROCEDURE opdracht9

(p\_dept\_id IN employees.department\_id%type

,p\_verhogingsperc IN number)

IS

v\_dept\_name departments.department\_name%type;

BEGIN

SELECT department\_name INTO v\_dept\_name

FROM departments

WHERE department\_id = p\_dept\_id;

DBMS\_OUTPUT.PUT\_LINE('Het gekozen departement is : ' || v\_dept\_name);

FOR lijn IN (SELECT last\_name, employee\_id, salary

FROM employees

WHERE department\_id = p\_dept\_id)

LOOP

DBMS\_OUTPUT.PUT\_LINE(lijn.employee\_id || ' ' || lijn.last\_name || ': '

|| lijn.salary);

END LOOP;

UPDATE employees SET salary = salary \* (1 + p\_verhogingsperc/100)

WHERE department\_id = p\_dept\_id;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Aantal salarisverhogingen : ' || sql%rowcount);

DBMS\_OUTPUT.PUT\_LINE('SITUATIE NA WIJZIGING.');

FOR lijn IN (SELECT last\_name, employee\_id, salary

FROM employees

WHERE department\_id = p\_dept\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(lijn.employee\_id || ' ' || lijn.last\_name || ': '

|| lijn.salary);

END LOOP;

END;

/

CREATE OR REPLACE PROCEDURE opdracht10

(p\_country\_id IN locations.country\_id%type

,p\_count\_loc OUT number)

AS

v\_country\_name countries.country\_name%type;

BEGIN

p\_count\_loc :=0;

FOR locrec IN (SELECT \*

FROM locations

WHERE country\_id = p\_country\_id

AND location\_id IN (select location\_id from departments))

LOOP

SELECT country\_name INTO v\_country\_name

FROM countries

WHERE country\_id = locrec.country\_id;

DBMS\_OUTPUT.PUT\_LINE('==> ' ||v\_country\_name ||' - ' ||locrec.location\_id || ' ' || locrec.city);

FOR deprec IN (SELECT department\_name, count(\*) aantal

FROM departments d join employees e using (department\_id)

WHERE d.location\_id = locrec.location\_id

GROUP BY department\_name)

LOOP

DBMS\_OUTPUT.PUT\_LINE(deprec.department\_name || ': ' || deprec.aantal);

p\_count\_loc := p\_count\_loc + deprec.aantal;

END LOOP;

END LOOP;

END opdracht8;

/

CREATE OR REPLACE TRIGGER bidr\_emp

BEFORE INSERT OR DELETE

ON employees

FOR EACH ROW

BEGIN

IF INSERTING THEN

:new.hire\_date := sysdate;

if to\_char(sysdate,'d') in (7,1) then

:new.hire\_date :=next\_day(sysdate,'mon');

end if;

:new.salary := 0;

ELSIF DELETING THEN

DELETE FROM job\_history

WHERE employee\_id = :old.employee\_id;

END IF;

END;

/

CREATE OR REPLACE TRIGGER bus\_jobs

BEFORE UPDATE OF min\_salary,max\_salary

ON jobs

BEGIN

IF USER IN('STUDENT','BEZOEKER') THEN

RAISE\_APPLICATION\_ERROR(-20000,'u heeft geen rechten voor deze actie');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Ben je zeker dat je het minimum of maximum slaries wilt aanpassen idien niet voer een rollbakc uit');

END IF;

END;

/

CREATE OR REPLACE TRIGGER aiud\_jhis

AFTER INSERT OR UPDATE OR DELETE

ON job\_history

DECLARE

v\_actie VARCHAR2(15) ;

BEGIN

IF INSERTING THEN

v\_actie := 'INSERT';

ELSIF UPDATING THEN

v\_actie := 'UPDATE';

ELSE

v\_actie := 'DELETE';

END IF;

INSERT INTO log\_history

VALUES(USER, SYSDATE, SYSTIMESTAMP, v\_actie);

END;

/

CREATE OR REPLACE TRIGGER bud\_is\_emp\_salary

BEFORE UPDATE OF salary OR DELETE OR INSERT

ON employees

DECLARE

v\_uur varchar2(8);

BEGIN

IF DELETING OR INSERING THEN

IF TO\_CHAR(SYSDATE,'D')=7 OR TO\_CHAR(SYSDATE,'D') =1 THEN

RAISE\_APPLICATION\_ERROR(-20000,'het is weekend');

END IF;

ELSE

v\_uur:=TO\_CHAR(SYSDATE,'hh24:mi:ss');

IF TOCHAR(SYSDATE,'d')!=2 OR v\_uur <'09:00:00' OR v\_uur > '17:00:00' THEN

RAISE\_APLICATION\_ERROR(-20000, 'salarisaanapssing nu niet mogelijk')

END IF;

END IF;

END;

/

CREATE OR REPLACE TRIGGER aur\_emp

AFTER UPDATE OF salary

ON employees

FOR EACH ROW

BEGIN

IF :NEW.salary < :OLD.salary THEN

RAISE\_APPLICATION\_ERROR(-20000,'Loonsverlaging kan niet!');

ELSIF :NEW.salary > :OLD.salary\*1.05 THEN

RAISE\_APPLICATION\_ERROR(-20000,'Een loonsverhoging van meer dan 5 % is niet toegelaten!');

END IF;

END;

CREATE OR REPLACE TRIGGER aur\_emp2

AFTER UPDATE OF salary

ON employees

FOR EACH ROW

WHEN (OLD.hire\_date < TO\_DATE('01-01-1995','dd-mm-yyyy'))

BEGIN

IF :NEW.salary < :OLD.salary THEN

RAISE\_APPLICATION\_ERROR(-20000,'Loonsverlaging kan niet!');

ELSIF :NEW.salary > :OLD.salary\*1.05 THEN

RAISE\_APPLICATION\_ERROR(-20000,'Een loonsverhoging van meer dan 5 % is niet toegelaten!');

END IF;

END;

CREATE OR REPLACE TRIGGER biur\_emp

BEFORE INSERT OR UPDATE

ON employees

FOR EACH ROW

BEGIN

:NEW.job\_id := UPPER(:NEW.job\_id);

:NEW.last\_name := INITCAP(:NEW.last\_name);

:NEW.first\_name := INITCAP(:NEW.first\_name);

IF INSERTING AND :NEW.hire\_date < SYSDATE THEN

RAISE\_APPLICATION\_ERROR (-20000,'Nieuwe aanwerfdatum kan niet in het verleden liggen!');

END IF;

IF :NEW.job\_id IN ('%MAN', '%MGR')

AND :OLD.job\_id NOT IN ('AD\_PRES','AD\_VP')

THEN

:NEW.salary := :OLD.salary \* 1.05;

END IF;

END biur\_emp;

CREATE OR REPLACE TRIGGER bidr\_emp

BEFORE INSERT OR DELETE ON employees

FOR EACH ROW

DECLARE

man departments.manager\_id%TYPE;

voornaam employees.first\_Name%TYPE;

naam employees.last\_name%TYPE;

BEGIN

IF INSERTING THEN

IF :NEW.manager\_id IS NULL THEN

SELECT manager\_id

INTO man

FROM departments

WHERE department\_id = :NEW.department\_id;

SELECT first\_name, last\_name

INTO voornaam, naam

FROM employees

WHERE employee\_id = man;

:NEW.manager\_id := man;

DBMS\_OUTPUT.PUT\_LINE('de chef wordt ' || voornaam || ' ' || naam);

END IF;

IF :NEW.hire\_date IS NULL THEN

:NEW.hire\_date := NEXT\_DAY(SYSDATE, 'MONDAY');

END IF;

IF :NEW.salary IS NULL THEN

:NEW.salary := 1000;

DBMS\_OUTPUT.PUT\_LINE('Het salaris van employee ' || :NEW.last\_name || ' wordt 1000.');

END IF;

ELSIF DELETING THEN

DBMS\_OUTPUT.PUT\_LINE('Dit is per maand een besparing van ' || :OLD.salary || ' euro.');

END IF;

END;

/