CORRECTION EXERCICEs CH3

Exercice 1

- **Yanis BELLAHCENE**

```
DO $$
DECLARE
V_SKIPPERS SKIPPERS%ROWTYPE;
BEGIN
SELECT s.* INTO V_SKIPPERS
FROM SKIPPERS s
JOIN CROISIERES c on c.SKNUM = s.SKNUM
WHERE c.CROISNUM='C001';
RAISE NOTICE 'Numéro: %', V_SKIPPERS.SKNUM;
RAISE NOTICE 'Nom: %', V_SKIPPERS.SKNOM;
RAISE NOTICE 'Port d''attache: %', V_SKIPPERS.SKPORT;
RAISE NOTICE 'Salaire: %', V_SKIPPERS.SALAIRE;
END $$;
```

- JAMHOUR Yousra

```
DO $$
DECLARE
Numeroskip skippers.sknum%type;
Nomskip skippers.sknom%type;
port_attache skippers.skport%type;
salairesk skippers.salaire%type;
BEGIN
SELECT sknum,sknom,skport,salaire  INTO
Numeroskip,Nomskip,port_attache,salairesk
FROM skippers natural join croisieres
WHERE croisnum='C001';
```

```
RAISE NOTICE 'Numero: %' , Numeroskip;

RAISE NOTICE 'nom %' , Nomskip;

RAISE NOTICE 'port dattache: %' , port_attache;

RAISE NOTICE 'salaire: %' ,salairesk;

END $$;
```

- **Korentin Georget**

Dans cette solution s'il y'a plusieurs ports qui sont associés au skipper, un seul port sera dans le résultat.

```
DO $$
DECLARE
V_SKIPPER SKIPPERS%ROWTYPE;
V_SKIPPER_PORTS SKIPPER_PORTS.skport%TYPE;
 BEGIN
SELECT s.sknum, s.sknom, s.salaire, sp.skport
INTO V_SKIPPER.sknum, V_SKIPPER.sknom, V_SKIPPER.salaire, V_SKIPPER_PORTS
FROM SKIPPERS s JOIN CROISIERES c USING (sknum) JOIN SKIPPER_PORTS sp USING (sknum)
WHERE c.croisnum = 'C001';
RAISE NOTICE 'Numéro: %' , V_SKIPPER.sknum;
RAISE NOTICE 'Nom: %' , V_SKIPPER.sknom;
RAISE NOTICE 'Port d''attache: %' , V_SKIPPER_PORTS;
RAISE NOTICE 'Salaire: %' , V_SKIPPER.salaire;
END $$;
```

Exercice 2 :

- JAMHOUR Yousra

```
DO $$

DECLARE

C CURSOR FOR

select distinct sknom
```

```
from skippers s natural join croisieres cr

where cr.depport='AJACCIO' ;

BEGIN

for ligne in C loop

RAISE NOTICE 'skipper au depart d ajaccio : %',ligne.sknom;

end loop;

END $$
```

- SANNA Thomas

```
DO $$

DECLARE

cu CURSOR FOR

SELECT sknom, salaire

FROM skippers

WHERE skport = 'AJACCIO';

nom skippers.sknom%TYPE;

sal skippers.salaire%TYPE;

BEGIN

OPEN cu;

LOOP
```

```
FETCH cu INTO nom, sal;

EXIT WHEN NOT FOUND;

RAISE NOTICE 'nom : %, salaire : %', nom, sal;

END LOOP;

CLOSE cu;

END;

$$;
```

Exercice 3:

- TOUIL Mohammed

```
create or replace FUNCTION MOYSAL() RETURNS INTEGER AS $$

DECLARE

MOY INTEGER;

BEGIN

SELECT AVG(salaire) INTO MOY

FROM SKIPPERS;

RETURN MOY;

END;

$$ LANGUAGE plpgsql;

DO $$

BEGIN

RAISE NOTICE 'Moyenne salaire : %', MOYSAL();

END $$;
```

- SANNA Thomas

```
CREATE OR REPLACE FUNCTION moysalville(ville skippers.skport%TYPE)
RETURNS double precision
LANGUAGE plpgsql
AS $$
DECLARE
moy double precision;
BEGIN
SELECT AVG(salaire)
INTO moy
FROM skippers
WHERE skport=ville;
RETURN moy;
END;
$$;
```

- **Yanis BELLAHCENE**

```
create or replace function MOYSAL(VILLE VARCHAR) RETURNS INTEGER AS $$
DECLARE
MOY INTEGER;
BEGIN
SELECT AVG(salaire) into MOY
from skippers
where skport = UPPER(VILLE);
return MOY;
END;
$$ LANGUAGE plpgsql;

DO $$
BEGIN
RAISE NOTICE 'Moyenne salaires : % euros', MOYSAL('Ajaccio');
```

END $$;

Exercice 3.3 - Définissez une procédure qui affiche les noms des skippers effectuant une croisière au départ d'une ville dont le nom est transmis en paramètre.

- SANNA Thomas

```
CREATE OR REPLACE PROCEDURE skipcroisville(IN ville skippers.skport%TYPE)

LANGUAGE plpgsql

AS $$

DECLARE

cu CURSOR FOR

SELECT sknom, salaire

FROM skippers

WHERE skport = ville;

BEGIN

FOR s IN cu LOOP

IF s IS NOT NULL THEN

RAISE NOTICE 'nom : %, salaire : %', s.sknom, s.salaire;

END IF;

END LOOP;

END;

$$;
```

Testez le fonctionnement de votre procédure en l'invoquant dans un bloc anonyme.

```
DO $$
```

```
BEGIN

CALL skipcroisville('AJACCIO');

END;

$$;
```

Exercice 4:

Résultat final : 3 5 6

EXERCICE 5 :

- **Yanis BELLAHCENE**

```
CREATE OR REPLACE FUNCTION verif_salaire()
RETURNS TRIGGER AS $$
BEGIN
        IF (NEW.SALAIRE< MOYSAL()) THEN
                NEW.SALAIRE := MOYSAL();
        END IF;
        RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER insert_update_skipper
BEFORE INSERT OR UPDATE OF SALAIRE ON SKIPPERS
FOR EACH ROW
EXECUTE FUNCTION verif_salaire();
```

E

Exercice 7

Dumenicu Franceschi

```
create table historique ( id serial primary key, nom_table varchar(255), type_maj
varchar(10), date_maj timestamp default current_timestamp );

create or replace function ajout_histo()

returns trigger as $$
```

```
begin
        insert into historique (nom_table, type_maj) values (TG_TABLE_NAME, TG_OP);
        return new;
end
$$ language plpgsql;
CREATE or replace TRIGGER add_log
AFTER INSERT OR UPDATE OR DELETE ON CROISIERES
FOR EACH ROW
EXECUTE FUNCTION ajout_histo();
```

- JAMHOUR Yousra

```
CREATE TABLE HISTORIQUE (
 id SERIAL PRIMARY KEY,
 nom_table VARCHAR(50),
type_maj VARCHAR(3) CHECK (type_maj IN ('INS', 'MOD', 'SUP')),
date_maj TIMESTAMP DEFAULT CURRENT_TIMESTAMP );
CREATE OR REPLACE FUNCTION historique_trigger()
RETURNS TRIGGER AS $$
BEGIN
 IF (TG_OP = 'INSERT') THEN
 INSERT INTO HISTORIQUE (nom_table, type_maj)
VALUES ('croisieres', 'INS');
ELSIF (TG_OP = 'UPDATE') THEN
    INSERT INTO HISTORIQUE (nom_table, type_maj)
    VALUES ('croisieres', 'MOD');
ELSIF (TG_OP = 'DELETE') THEN
```

```
    INSERT INTO HISTORIQUE (nom_table, type_maj)
    VALUES ('croisieres', 'SUP');
END IF;

RETURN NULL;


END;

 $$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_maj_croisieres

AFTER INSERT OR UPDATE OR DELETE ON croisieres

FOR EACH ROW EXECUTE FUNCTION historique_trigger();

INSERT INTO CROISIERES (CROISNUM, SKNUM, BATNUM, DEPPORT, ARRPORT,
DEPDATE, ARRDATE)

VALUES ('C011', '1', 'B002', 'BASTIA', 'CALVI', DATE '2024-07-10', DATE '2024-07-11');
```