Session Date	:	4 Oct 2021
Semester	:	3
Subject	:	Basis Data
Topic	:	Function in SQL Server (PL/SQL)
Activity	:	Practicing Programming (PL/SQL) in SQL Server
Duration	:	110 minutes
Rules		Individual
Deliverable	:	Softcopy
Dead line	:	End of Session
Place to deliver	:	http://ecourse.del.ac.id/
Objective	:	Students able to use PL/SQL in SQL Server
Lecturer	:	PAT/IUS/RSL

Execute sql code below to create a product table

```
CREATE TABLE product (
    prod_nr INT NOT NULL
    CONSTRAINT pk_product PRIMARY KEY (prod_nr),
    Name VARCHAR (30) NOT NULL,
    Price MONEY NOT NULL,
    Type VARCHAR (30) NOT NULL
)

INSERT product (prod_nr, name, price, type)
VALUES (1, 'tv', 500, 'electronics');

INSERT product (prod_nr, name, price, type)
VALUES (2, 'radio', 100, 'electronics');

INSERT product (prod_nr, name, price, type)
VALUES (3, 'ball', 100, 'sports');

INSERT product (prod_nr, name, price, type)
VALUES (4, 'racket', 200, 'sports');
```

Result:

	prod_nr	Name	Price	Type
1	1	tv	500,00	electronics
2	2	radio	100,00	electronics
3	3	ball	100,00	sports
4	4	racket	200,00	sports

Exercise 1

Create a function with an input parameter the name of the product. Based on this input, the function should return or print a message like this: 'There are (**the name of the product**) in stock' or 'There are no (**the name of the product**) in stock'.

Example of Result:

```
-- Execute
SELECT [dbo].[fn_stok] ('book');
SELECT [dbo].[fn_stok] ('TV');
```

There are NObookin stock → **Karena tidak ada book dalam table product**

There are TV in stock → **Karena ada TV dalam table product**

Exercise 2

Create a function with a numeric input parameter. Based on this input, the function should return or print a message like this: 'the average price of sport products is greater or equal or less than (**the value of the input**)' when that is the case in the database.

Example of Result:

```
-- Execute
SELECT AVG (Price) AS AVG_PRICE
FROM product
WHERE Type ='sports'

SELECT [dbo].[fng_avg_price_sport] (100);
SELECT [dbo].[fng_avg_price_sport] (150);
SELECT [dbo].[fng_avg_price_sport] (400);
```

	AVG_PRICE
1	150,00

	(No column name)
1	The average price of sports product is greater than 100

	(No column name)
1	The average price of sports products is equal 150

	(No column name)
1	The average price of sports products is less than 400

Exercise 3

Create a function to update the price of all records in table product by 10% until the average price is greater than 500 (**500 is defined by user through an input parameter**). **Hint**: You have to return a table.

Result:

	prod_nr	Name	Price	Type
1	1	tv	500,00	electronics
2	2	radio	100,00	electronics
3	3	ball	100,00	sports
4	4	racket	200,00	sports

	AVG_PRICE
1	225,00

	prod_id	price
1	1	1175
2	2	233
3	3	233
4	4	468

	AVG_PRICE
1	527

Exercise 4

Create a function **CheckModulo11**, that checks if a given **accountNr** is a valid number.
Ex. 972428577 is valid, because:
9*9+8*7+7*2+6*4+5*2+4*8+3*5+2*7+1*7)%11 =0 Use this function in a check clause in a table with a column that represents an **accountNr**.
You need to create a table to store the accountNr by adding constraint and try to insert data into this table.

Result:

```
Print DBO.fnCheckModulo11 (972428577)

valid
```

Exercise 5

Create a function with heading:

```
CREATE FUNCTION fnTableSundays
    (
        @dateFrom DATETIME,
        @dateTo DATETIME)

    RETURNS @tblSunday TABLE (nummer SMALLINT, date DATETIME)
```

Which returns the dates of all sundaysbetween @dateFrom and @dateTo in a table with columns number and Sunday.

Result:

```
SELECT * FROM dbo.fnTableSundays('2008-03-08', '2008-05-09');
```

	nummer	date
1	1	2008-03-09 00:00:00.000
2	2	2008-03-16 00:00:00.000
3	3	2008-03-23 00:00:00.000
4	4	2008-03-30 00:00:00.000
5	5	2008-04-06 00:00:00.000
6	6	2008-04-13 00:00:00.000
7	7	2008-04-20 00:00:00.000
8	8	2008-04-27 00:00:00.000
9	9	2008-05-04 00:00:00.000

Selamat mengerjakan