# Transact-SQL – ćwiczenia.

*Procedury, Funkcje.*

Utwórz **procedurę** p\_z1 (bez parametrów), która wyświetli nazwę produktu (ProductName) oraz cenę jednostkową (UnitPrice) z tabeli Products. Wszystkie rekordy powinny zostać wyświetlone. Wykonaj procedurę.

CREATE PROCEDURE P\_Z1

AS

SELECT ...

EXEC P\_Z1

create proc p\_z1

as

select ProductName, UnitPrice

from Products;

go

exec p\_z1;

Utwórz **funkcję** f\_z2 (bez parametrów), która wyświetli nazwę produktu (ProductName) oraz cenę jednostkową (UnitPrice) z tabeli Products. Wszystkie rekordy powinny zostać wyświetlone. Sprawdź działanie funkcji.

CREATE FUNCTION F\_Z2()

RETURNS TABLE

AS

RETURN ( ... )

SELECT \* FROM F\_Z2()

create function f\_z2()

returns table

as

return (

select ProductName, UnitPrice

from Products

);

go

select \* from f\_z2();

Utwórz **procedurę** p\_z3, która wyświetli nazwę produktu (ProductName) oraz cenę jednostkową (UnitPrice) z tabeli Products. Tym razem jednak procedura powinna wyświetlić dane tylko tych produktów, których cena jednostkowa jest większa niż wartość parametru procedury. Wykonaj procedurę.

CREATE PROCEDURE P\_Z3

(@Price MONEY)

AS

SELECT ...

Wykonaj procedurę:

ZEC P\_Z3 50.0

ZEC P\_Z3 90.0

create proc p\_z3 @price money

as

select ProductName, UnitPrice

from Products

where UnitPrice > @price;

go

exec p\_z3 50.00;

exec p\_z3 90.00;

Utwórz **funkcję** f\_z4, która wyświetli nazwę produktu (ProductName) oraz cenę jednostkową (UnitPrice) z tabeli Products. Tym razem jednak funkcja powinna wyświetlić dane tylko tych produktów, których cena jednostkowa jest większa niż wartość parametru funkcji. Sprawdź działanie funkcji.

Wywołanie funkcji:

SELECT \* FROM F\_Z4(50)

SELECT \* FROM F\_Z4(90)

create function f\_z4(@price money)

returns table

as

return (

select ProductName, UnitPrice

from Products

where UnitPrice > @price

);

go

select \* from f\_z4(50);

select \* from f\_z4(90);

Utwórz **procedurę** p\_z5, która wyświetli ProductName, UnitPrice oraz CategoryID dla wszystkich produktów, które należą do kategorii o **nazwie** (CategoryName) podanej jako parametr procedury. Nazwy kategorii są unikalne i można je odnaleźć w tabeli Categories (kolumna CategoryName).

Wykonaj procedurę.

create proc p\_z5 @name nvarchar(30)

as

select p.ProductName, p.UnitPrice, p.CategoryID

from Products as p

inner join Categories as c

on p.CategoryID = c.CategoryID

where c.CategoryName = @name;

go

exec p\_z5 N'Beverages';

Utwórz **funkcję** f\_z6, która wyświetli ProductName, UnitPrice oraz CategoryID dla wszystkich produktów, które należą do kategorii o **nazwie** (CategoryName) podanej jako parametr procedury. Nazwy kategorii są unikalne i można je odnaleźć w tabeli Categories (kolumna CategoryName).

Sprawdź działanie funkcji.

create function f\_z6(@name nvarchar(30))

returns table

as

return (

select p.ProductName, p.UnitPrice, p.CategoryID

from Products as p

inner join Categories as c

on p.CategoryID = c.CategoryID

where c.CategoryName = @name

);

go

select \* from f\_z6(N'Beverages');

Utwórz **funkcję** f\_z7, która wyświetli ProductID, ProductName, CategoryID, UnitPrice wszystkich produktów o cenach zawierających się przedziale wyznaczonym przez parametry funkcji.

create function f\_z7(

@min money,

@max money

)

returns table

as

return (

select ProductID, ProductName, UnitPrice, CategoryID

from Products

where UnitPrice between @min and @max

);

go

select \* from f\_z7(10, 20);

Utwórz **funkcję** f\_z8. Skopiuj tekst funkcji f\_z7 i dodaj jeden więcej parametr – @CategoryName. Funkcja powinna wyświetlić wszystkie dane produktów o cenach zawierających się przedziale wyznaczonym przez pierwsze dwa parametry funkcji. Ponadto rezultat powinien być ograniczony tylko do produktów z kategorii, której **nazwa** podana jest jako trzeci parametr funkcji.

create function f\_z8(

@min money,

@max money,

@name nvarchar(30)

)

returns table

as

return (

select p.ProductID, p.ProductName, p.UnitPrice, c.CategoryName

from Products as p

inner join Categories as c

on p.CategoryID = c.CategoryID

where UnitPrice between @min and @max

and c.CategoryName = @name

);

go

select \* from f\_z8(10, 20, N'Beverages');

Utwórz **funkcję** f\_z9. Skopiuj tekst funkcji f\_z8. Zmień funkcję f\_z9 tak, żeby sprawdzała czy parametr @CategoryName jest równy NULL. Jeśli jest, to funkcja powinna wyświetlić dane produktów wszystkich kategorii.

Wskazówka:

CREATE FUNCTION f\_z11 (@Price1 MONEY, @Price2 MONEY, @CategoryID INT)

RETURNS @T TABLE (ProductID INT, ProductName NVARCHAR(40),

CategoryID INT,

UnitPrice MONEY)

AS

BEGIN

IF @CategoryID IS NULL

INSERT INTO @T

SELECT ProductID, ProductName, CategoryID

FROM Products

ELSE

INSERT INTO @T

SELECT ProductID, ProductName, CategoryID

FROM Products

WHERE ....

RETURN

END

create function f\_z9(

@min money,

@max money,

@name nvarchar(15)

)

returns @temp table (

ProductID int,

ProductName nvarchar(40),

UnitPrice money,

CategoryID int

)

as

begin

if @name is null

insert into @temp

select ProductID,

ProductName,

UnitPrice,

CategoryID

from Products

where UnitPrice between @min and @max

else

insert into @temp

select p.ProductID,

p.ProductName,

p.UnitPrice,

p.CategoryID

from Products as p

inner join Categories as c

on p.CategoryID = c.CategoryID

where UnitPrice between @min and @max

and c.CategoryName = @name

return

end

go

select \* from f\_z9(10, 20, N'Beverages');

select \* from f\_z9(10, 20, null);

# – nieobowiązkowe.

Utwórz **funkcję** f\_z10, która wyświetli Numer zamówienia i Datę zamówienia (OrderId, OrderDate) trzech najnowszych zamówień (Orders, WITH TIES) trzech najlepszych klientów (WITH TIES). Najlepszy klient to taki klient, który wydał najwięcej pieniędzy (na wszystkich zamówieniach). Wzór obliczenia kwoty za jedną linię zamówienia w tabeli [Order Details] to ROUND(Quantity\*UnitPrice\*CAST((1-Discount) AS MONEY),2).

create function f\_z10()

returns table

as

return (

select top 3 with ties o.OrderID, cast(o.OrderDate as date) as OrderDate

from Orders as o

inner join (

select top 3 with ties o.CustomerID,

sum(round(od.Quantity \* od.UnitPrice \* cast((1 - od.Discount) as money), 2)) as TotalSum

from [Order Details] as od

inner join Orders as o

on od.OrderID = o.OrderID

group by o.CustomerID

order by TotalSum desc

) as c

on o.CustomerID = c.CustomerID

order by OrderDate desc

);

go

select \* from f\_z10();