ĆW 4

-- 1. Napisz prosty program w Transact-SQL.  
-- Zadeklaruj zmienną, przypisz na tą zmienną liczbę rekordów w tabeli Emp (lub jakiejkolwiek innej)  
-- i wypisz uzyskany wynik używając instrukcji PRINT, w postaci napisu np. "W tabeli jest 10 osób".  
  
DECLARE @zmienna int = (SELECT COUNT(\*) FROM emp)  
PRINT 'W tabeli jest ' + cast(@zmienna AS varchar) + ' osób';  
  
-- 2. Używając Transact-SQL, policz liczbę pracowników z tabeli EMP.  
-- Jeśli liczba jest mniejsza niż 16, wstaw pracownika Kowalskiego i wypisz komunikat.  
-- W przeciwnym przypadku wypisz komunikat informujący o tym, że nie wstawiono danych.  
  
DECLARE @zmienna int  
SELECT @zmienna= (SELECT COUNT(\*) FROM emp)  
IF @zmienna < 16  
 BEGIN  
 INSERT INTO emp(empno, ename, job, mgr, hiredate, sal, comm, deptno)  
 VALUES(7908,'Jan','Kowalski',null, '1980-12-17 00:00:00.000', 2000,10,10);  
 PRINT 'Wstawiono Kowalskiego';  
 END  
ELSE  
PRINT 'Nie wstawiono danych';  
  
-- 3. Napisz procedurę zwracającą pracowników, którzy zarabiają więcej niż wartość zadana parametrem procedury.  
CREATE PROCEDURE *listaPracownikow* @sal int =2000, @imie varchar(20)  
AS  
 BEGIN  
 SELECT sal, @imie=emp.ENAME FROM emp  
 WHERE sal > @sal;  
 PRINT @imie;  
 END;  
  
---- odpalamy osobno  
EXECUTE *listapracownikow* 1000;  
  
-- 4. Napisz procedurę służącą do wstawiania działów do tabeli Dept.  
-- Procedura będzie pobierać jako parametry: nr\_działu, nazwę i lokalizację.  
-- Należy sprawdzić, czy dział o takiej nazwie lub lokalizacji już istnieje.  
-- Jeżeli istnieje, to nie wstawiamy nowego rekordu.  
CREATE PROCEDURE *insertDept* @nr\_dzialu int, @nazwa varchar(20), @lokalizacja varchar(20)  
AS BEGIN  
 IF NOT EXISTS(SELECT 'X' FROM DEPT WHERE @nazwa = DNAME OR @lokalizacja = LOC)  
 BEGIN  
 INSERT INTO DEPT(DEPTNO, DNAME, LOC)  
 VALUES (@nr\_dzialu, @nazwa, @lokalizacja);  
 PRINT 'Wstawiono';  
 END  
 ELSE  
 print 'Nie wstawiono';  
END;  
  
-- 5. Napisz procedurę umożliwiającą użytkownikowi wprowadzanie nowych pracowników do tabeli EMP.  
-- Jako parametry będziemy podawać nazwisko i nr działu zatrudnianego pracownika.  
-- Procedura powinna wprowadzając nowy rekord sprawdzić, czy wprowadzany dział istnieje  
-- (jeżeli nie, to należy zgłosić błąd) oraz obliczyć mu pensję równą minimalnemu zarobkowi w tym dziale.  
-- EMPNO nowego pracownika powinno zostać wyliczone jako najwyższa istniejąca wartość w tabeli + 1.  
CREATE PROCEDURE *newEmploye* @nazwisko varchar(20), @nr\_dzialu int  
AS BEGIN  
 IF EXISTS(SELECT 'X' FROM EMP WHERE @nr\_dzialu = DEPTNO)  
 BEGIN  
 DECLARE @pensja INT = (SELECT(MIN(sal)) FROM EMP WHERE DEPTNO = @nr\_dzialu GROUP BY DEPTNO);  
 INSERT INTO EMP(EMPNO, ENAME, SAL, DEPTNO)  
 SELECT MAX(EMPNO)+1, @nazwisko, @pensja, @nr\_dzialu FROM EMP;  
 END  
 ELSE  
 print 'Dzial nie istnieje';  
END;

ĆW 5

--- 1.Przy pomocy kursora przejrzyj wszystkich pracowników i zmodyfikuj wynagrodzenia tak, aby osoby zarabiające mniej niż 1000  
-- miały zwiększone wynagrodzenie o 10%, natomiast osoby zarabiające powyżej 1500 miały zmniejszone wynagrodzenie o 10%.  
-- Wypisz na ekran każdą wprowadzoną zmianę.  
DECLARE kursor CURSOR FOR  
SELECT empno, ename, sal FROM EMP WHERE sal < 1000 OR sal > 1500  
DECLARE @empno int, @ename varchar(20), @sal DECIMAL(8,2)  
OPEN kursor  
FETCH NEXT FROM kursor INTO @empno, @ename, @sal  
WHILE @@FETCH\_STATUS=0  
BEGIN  
 IF @sal <1000  
 BEGIN  
 SET @sal=@sal\*1.1  
 UPDATE emp SET sal=@sal WHERE empno=@empno  
 PRINT 'Zwiększono pensje o 10% pracownikowi '+ @ename+' o '+ CAST(@sal as varchar);  
 END  
 ELSE IF @sal>1500  
 BEGIN  
 SET @sal=@sal\*0.9  
 UPDATE emp SET sal=@sal WHERE empno=@empno  
 PRINT 'Zwiększono pensje o 10% pracownikowi '+ @ename+' o '+ CAST(@sal as varchar);  
 END  
 FETCH NEXT FROM kursor INTO @empno,@ename,@sal  
END  
CLOSE kursor  
DEALLOCATE kursor;  
  
--- 2. Przerób kod z zadania 1 na procedurę tak, aby wartości zarobków (1000 i 1500) nie były stałe, tylko były parametrami procedury.  
DECLARE kursor CURSOR FOR  
SELECT empno, ename, sal FROM EMP  
DECLARE @empno int, @ename varchar(20), @sal DECIMAL(8,2), @parametr1 int = 1000, @parametr2 int = 1500  
OPEN kursor  
FETCH NEXT FROM kursor INTO @empno, @ename, @sal  
WHILE @@FETCH\_STATUS=0  
BEGIN  
 IF @sal < @parametr1  
 BEGIN  
 SET @sal=@sal\*1.1  
 UPDATE emp SET sal=@sal WHERE empno=@empno  
 PRINT 'Zwiększono pensje o 10% pracownikowi '+ @ename+' o '+ CAST(@sal as varchar);  
 END  
 IF @sal > @parametr2  
 BEGIN  
 SET @sal=@sal\*0.9  
 UPDATE emp SET sal=@sal WHERE empno=@empno  
 PRINT 'Zwiększono pensje o 10% pracownikowi '+ @ename+' o '+ CAST(@sal as varchar);  
  
 END  
 FETCH NEXT FROM kursor INTO @empno,@ename,@sal  
END  
CLOSE kursor  
DEALLOCATE kursor;  
  
--- 3. W procedurze sprawdź średnią wartość zarobków z tabeli EMP z działu określonego parametrem procedury.  
-- Następnie należy dać prowizję (comm) tym pracownikom tego działu, którzy zarabiają poniżej średniej.  
-- Prowizja powinna wynosić 5% ich miesięcznego wynagrodzenia.  
CREATE PROCEDURE *prowizja* @deptno INT  
AS BEGIN  
 UPDATE emp  
 SET comm = sal \* 0.05  
 WHERE sal < (SELECT AVG(sal) FROM emp WHERE deptno = @deptno)  
 AND deptno = @deptno  
end  
  
--- 4. (bez kursora) Utwórz tabelę Magazyn (IdPozycji, Nazwa, Ilosc) zawierającą ilości poszczególnych towarów w magazynie  
-- i wstaw do niej kilka przykładowych rekordów. W bloku Transact-SQL sprawdź, którego artykułu jest najwięcej w magazynie  
-- i zmniejsz ilość tego artykułu o 5 (jeśli stan jest większy lub równy 5, w przeciwnym wypadku zgłoś błąd).  
CREATE TABLE Magazyn(Idpozycji int PRIMARY KEY,nazwa varchar(10), ilosc int)  
INSERT INTO Magazyn (Idpozycji, nazwa, ilosc) VALUES ((SELECT MAX(Idpozycji) FROM Magazyn) + 1, 'Ksiażka', 4)  
INSERT INTO Magazyn (Idpozycji, nazwa, ilosc) VALUES ((SELECT MAX(Idpozycji) FROM Magazyn) + 1, 'Kubek', 15)  
INSERT INTO Magazyn (Idpozycji, nazwa, ilosc) VALUES ((SELECT MAX(Idpozycji) FROM Magazyn) + 1, 'Plecak', 11)  
  
DECLARE @zmienna int  
SELECT @zmienna= (SELECT MAX(ilosc) FROM Magazyn);  
IF @zmienna >=5  
 BEGIN  
 UPDATE Magazyn SET ilosc=ilosc-5 WHERE ilosc=@zmienna;  
 end  
ELSE  
PRINT 'Błąd';  
  
--- 5. Przerób kod z zadania 4 na procedurę, której będziemy mogli podać wartość, o którą zmniejszamy stan (zamiast wpisanego „na sztywno” 5).  
CREATE PROCEDURE *wstaw* @zmienna int  
AS BEGIN  
DECLARE @zmienna1 int  
SELECT @zmienna1= (SELECT MAX(ilosc) FROM Magazyn);  
IF @zmienna1 >=5  
 BEGIN  
 UPDATE Magazyn SET ilosc=ilosc-@zmienna WHERE ilosc=@zmienna1;  
 end  
ELSE  
PRINT 'Błąd'  
END;

ĆW 6

--1. Utwórz wyzwalacz, który nie pozwoli usunąć rekordu z tabeli Emp.  
CREATE TRIGGER TR1  
ON EMP  
FOR DELETE  
AS  
ROLLBACK;  
--ENABLE/DISABLE TRIGGER TR1 ON EMP;  
ENABLE TRIGGER TR1 on emp;  
---2.  
-- Utwórz wyzwalacz, który przy wstawianiu pracownika do tabeli Emp,  
-- wstawi prowizję równą 0, jeśli prowizja była pusta.  
-- Uwaga: Zadanie da się wykonać bez użycia wyzwalaczy przy pomocy DEFAULT.  
-- Użyjmy jednak wyzwalacza w celach treningowych.  
CREATE TRIGGER TR2  
ON EMP FOR INSERT AS  
BEGIN  
 DECLARE @comm int, @empno int  
 SELECT @comm=comm, @empno=empno FROM inserted;  
 if @comm is null  
 BEGIN  
 SET @comm=0  
 UPDATE emp SET comm=@comm WHERE empno=@empno  
 end  
end  
CREATE TRIGGER TR2  
 ON EMP  
 FOR INSERT  
 AS BEGIN  
 UPDATE EMP  
 SET comm=0  
 WHERE comm is null  
 AND empno IN (SELECT empno FROM inserted)  
end  
ENABLE TRIGGER TR2 ON emp;  
INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)  
VALUES(123,'Karolina','ok', null,'2020-12-02 00:00:00.000',1001,null,10)  
SELECT \* FROM emp;  
DELETE emp where ename like 'Karolina';  
  
---3.Utwórz wyzwalacz, który przy wstawianiu lub modyfikowaniu danych w tabeli Emp sprawdzi czy nowe zarobki  
-- (wstawiane lub modyfikowane) są większe niż 1000.  
-- W przeciwnym przypadku wyzwalacz powinien zgłosić błąd i nie dopuścić do wstawienia rekordu.  
-- Uwaga: Ten sam efekt można uzyskać łatwiej przy pomocy więzów spójności typu CHECK.  
-- Użyjmy wyzwalacza w celach treningowych.  
CREATE TRIGGER TR3  
 ON EMP FOR INSERT,UPDATE AS  
 BEGIN  
 DECLARE @sal int, @empno int  
 SELECT @sal=sal, @empno=empno FROM inserted;  
 if EXISTS (SELECT sal FROM inserted WHERE sal<1000)  
 BEGIN  
 ROLLBACK;  
 RAISERROR ('Niedopuszczalna wartość SAL!',1,2);  
 END  
END  
ENABLE TRIGGER TR3 ON emp;  
INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)  
VALUES(123,'Karolina','ok', null,'2020-12-02 00:00:00.000',1001,null,10)  
SELECT \* FROM emp;  
DELETE emp where ename like 'Karolina';  
  
--4. Utwórz tabelę budzet:  
CREATE TABLE budzet (wartosc INT NOT NULL)  
--W tabeli tej będzie przechowywana łączna wartość wynagrodzenia wszystkich pracowników.  
-- Tabela będzie zawsze zawierała jeden wiersz.  
-- Należy najpierw obliczyć początkową wartość zarobków:  
INSERT INTO budzet (wartosc)  
SELECT SUM(sal) FROM emp  
--Utwórz wyzwalacz, który będzie pilnował, aby wartość w tabeli budzet była zawsze aktualna,  
-- a więc przy wszystkich operacjach aktualizujących tabelę emp (INSERT, UPDATE, DELETE),  
-- wyzwalacz będzie aktualizował wpis w tabeli budżet  
CREATE TRIGGER TR4  
 ON EMP FOR INSERT,UPDATE, DELETE AS  
BEGIN  
 DECLARE @wartosc int  
 BEGIN  
 UPDATE budzet SET wartosc=(SELECT SUM(sal) FROM emp) WHERE wartosc=@wartosc;  
 END  
END;  
ENABLE TRIGGER TR4 ON emp;  
SELECT \*FROM budzet;  
  
---5.Napisz wyzwalacz, który nie pozwoli modyfikować nazw działów w tabeli dept.  
-- Powinno być jednak możliwe wstawianie nowych działów.  
CREATE TRIGGER TR5  
 ON DEPT FOR UPDATE AS ROLLBACK;  
ENABLE TRIGGER TR5 ON dept;  
UPDATE dept set dname='SALES' where dname='tak';  
INSERT INTO dept (DEPTNO, DNAME, LOC) VALUES (60,'ok','tak');  
DELETE dept WHERE dname='ok';  
SELECT \* FROM DEpt;  
  
--6. Napisz jeden wyzwalacz, który:  
--• Nie pozwoli usunąć pracownika, którego pensja jest większa od 0.  
--• Nie pozwoli zmienić nazwiska pracownika.  
--• Nie pozwoli wstawić pracownika, który już istnieje (sprawdzając po nazwisku).  
 CREATE TRIGGER TR6  
 ON EMP FOR INSERT,UPDATE,DELETE AS  
BEGIN  
 DECLARE @sal int, @empno int, @ename varchar  
 SELECT @empno=empno, @ename=ename FROM inserted;  
 SELECT @sal=sal FROM deleted  
 IF @sal<=0  
 BEGIN  
 ROLLBACK;  
 END  
 IF @ename IN (SELECT @ename FROM EMP)  
 BEGIN  
 ROLLBACK;  
 END  
 IF UPDATE(ename)  
 BEGIN  
 ROLLBACK;  
 END  
END  
ENABLE TRIGGER TR6 on emp;  
DELETE emp WHERE sal=1000;  
SELECT \* FROM emp;  
ROLLBACK;  
--7. Napisz wyzwalacz, który:  
--• Nie pozwoli zmniejszać pensji.  
--• Nie pozwoli usuwać pracowników.  
  
  
DISABLE TRIGGER TR1 on emp;  
DISABLE TRIGGER TR2 on emp;  
DISABLE TRIGGER TR3 on emp;  
DISABLE TRIGGER TR4 on emp;  
DISABLE TRIGGER TR5 on emp;  
DISABLE TRIGGER TR6 on emp;