Dokumentacja Projektu Bazy Danych

Baza Danych dla Klubu Rozrywki

Wykonali:

Marcel Zmeczerowski & Jerzy Recław

# Opis Bazy Danych:

Projekt przedstawia bazę danych klubu rozrywki, w którym pracują pracownicy na różnych stanowiskach. Mogą brać urlopy i mieć doliczane premie do ich wypłat. Klub ten organizuje imprezy, gdzie występują różni artyści. Do danej imprezy przypisywani są pracownicy kluby. Przychodzący klienci mogą składać zamówienia u kasjerów obsługujących kasy fiskalne. Zamówić można albo produkt należący do pewnej kategorii produktu i dostarczony od danego dostawcy, albo usługę sprzedawaną od wolnych stanowisk, gdzie można z tych usług skorzystać. Przykładowymi produktami mogą być różnego rodzaje alkohole, napoje bezalkoholowe, albo drobne przekąski. Dostępne usługi to np. bilard, piłkarzyki czy rzutki.

# Użyte funkcje w Projekcie:

|  |  |  |
| --- | --- | --- |
| Nr. | Opis | Kod |
| 1 | ilość produktu w magazynie - stan(produkt) | CREATE FUNCTION stan(@nazwaproduktu VARCHAR(30))  RETURNS INT  BEGIN  RETURN (SELECT [stan\_magazyn(w kg)] FROM Produkty WHERE nazwa\_produktu = @nazwaproduktu)  END |
| 2 | Funkcja sprawdzająca czy jest wolne stanowisko do np. gry w bilard | CREATE FUNCTION czy\_wolne\_stanowisko(@id\_usluga INT)  RETURNS VARCHAR(50)  AS  BEGIN  DECLARE @Wynik VARCHAR(50)  IF EXISTS (SELECT czy\_wolne FROM StanowiskaUslug WHERE id\_usluga=@id\_usluga AND czy\_wolne = 1 )  SET @Wynik='Jest wolne stanowisko'  ELSE  SET @Wynik='Nie ma wolnego stanowiska'  RETURN @Wynik  END |
| 3 | Funkcja wypisująca dane  wypłat z danego dnia. | CREATE FUNCTION wyplaty(@data DATE)  RETURNS TABLE AS  RETURN (SELECT \* FROM Wyplata WHERE data\_wyplaty = @data) |
| 4 | Funkcja wypisująca dane pracowników mających więcej pieniędzy na godzinę niż podana kwota. | CREATE FUNCTION zarabiaja\_wiecej\_niz(@placa MONEY)  RETURNS TABLE  AS  RETURN  (SELECT P.imie,P.nazwisko,S.stawka\_godzinowa,S.nazwa\_stanowiska FROM Pracownicy P JOIN StanowiskaPracy S ON P.id\_stanowisko=S.id\_stanowisko  WHERE S.stawka\_godzinowa > @placa) |
| 5 | Funkcja zwracająca jacy pracownicy byli na imprezie. | CREATE FUNCTION kto\_na\_imprezie(@id\_imprezy int)  RETURNS TABLE AS  RETURN (SELECT p.imie, p.nazwisko FROM Pracownicy as p  join Impreza\_pracownicy as imp ON p.id\_pracownik = imp.id\_pracownik where imp.id\_impreza = @id\_imprezy) |
| 6 | Funkcja wypisująca ilość zamówień danego produktu w danym dniu. | CREATE FUNCTION ilosc\_w\_danym\_dniu(@id\_produktu INT, @Data DATE )  RETURNS INT  AS  BEGIN  DECLARE @Ilosc INT  SET @Ilosc = (SELECT SUM(ilosc) FROM Zamowienia WHERE id\_produkt = @id\_produktu AND (CAST(data\_zamowienia AS DATE)) = @Data)  RETURN @Ilosc  END |
| 7 | Funkcja wypisująca produkty od danego dostawcy. | CREATE FUNCTION produkty\_dostawcy(@id\_dostawcy INT)  RETURNS TABLE AS  RETURN (SELECT \* FROM Produkty  WHERE id\_dostawca = @id\_dostawcy) |
| 8 | Funkcja wypisująca pracowników mających kiedykolwiek dany rodzaj premii. | CREATE FUNCTION kto\_mial\_premie(@id\_premia INT)  RETURNS TABLE  AS  RETURN (  SELECT P.imie,P.nazwisko,W.data\_wyplaty,PR.id\_premii,PR.opis FROM Pracownicy P  JOIN Wyplata W ON P.id\_pracownik = W.id\_pracownik  JOIN Premia PR ON W.id\_premia = PR.id\_premii  WHERE PR.id\_premii = @id\_premia) |
| 9 | Funkcja wypisująca pracowników którzy w danym momencie są dostępni (bez urlopu). | CREATE FUNCTION dostepni\_pracownicy()  RETURNS TABLE AS  RETURN (SELECT imie, nazwisko  FROM Pracownicy  WHERE id\_pracownik NOT IN (SELECT id\_pracownik FROM Urlopy where od\_kiedy <= getdate() and do\_kiedy >=getdate())) |
| 10 | Funkcja wypisuje dane pracowników będących na imprezie danego artysty. | CREATE FUNCTION Artysta\_a\_pracownicy (@id\_artysta INT)  RETURNS TABLE  AS  RETURN (  SELECT P.\* FROM Pracownicy P JOIN Impreza\_pracownicy IMP ON P.id\_pracownik = IMP.id\_pracownik  JOIN Impreza I ON IMP.id\_impreza = I.id\_impreza  JOIN Impreza\_artysci IA ON I.id\_impreza = IA.id\_artysta  WHERE IA.id\_artysta = @id\_artysta) |
| 11 | Funkcja wypisująca ile jest pracowników na danym stanowisku. | CREATE FUNCTION zapelnienie\_stanowiska\_pracy(@id\_stanowisko INT)  RETURNS INT  AS  BEGIN  RETURN (SELECT COUNT(\*) FROM Pracownicy WHERE id\_stanowisko = @id\_stanowisko)  END |
| 12 | Funkcja wypisująca na ilu imprezach był dany artysta. | CREATE FUNCTION Na\_ilu\_imprezach (@id\_artysta INT)  RETURNS INT  AS  BEGIN  RETURN (SELECT COUNT(\*) FROM Impreza\_artysci WHERE id\_artysta = @id\_artysta)  END |
| 13 | Funkcja zwracająca aktualny stan łączny wszystkich kas. | CREATE FUNCTION stan\_kas()  RETURNS FLOAT  AS  BEGIN  RETURN (select SUM(stan) from Kasa)  END |
| 14 | Funkcja zwracająca dzienny obrót na danej kasie | CREATE FUNCTION obrot\_kasy(@id\_kasa INT, @data DATE)  RETURNS FLOAT  AS  BEGIN  DECLARE @Ilosc\_usluga FLOAT  DECLARE @Ilosc\_produkt FLOAT  SET @Ilosc\_produkt = (SELECT SUM( Z.ilosc \* P.cena\_za\_porcje)  Sumka FROM Zamowienia Z JOIN Produkty P ON Z.id\_produkt=P.id\_produkt  WHERE Z.id\_kasa= @id\_kasa AND CAST(Z.data\_zamowienia AS DATE) = @data)  IF @Ilosc\_produkt IS NULL  SET @Ilosc\_produkt = 0  SET @Ilosc\_usluga = (SELECT SUM(Z.ilosc \* U.cena)  Sumka FROM Zamowienia Z JOIN StanowiskaUslug SU ON Z.id\_stanowisko=SU.id\_stanowisko  JOIN Usługa U On SU.id\_usluga= U.id\_usluga  WHERE Z.id\_kasa= @id\_kasa AND CAST(Z.data\_zamowienia AS DATE) = @data)  IF @Ilosc\_usluga IS NULL  SET @Ilosc\_usluga = 0  RETURN @Ilosc\_usluga + @Ilosc\_produkt  END |
| 15 | Funkcja licząca który pracownik najwięcej sprzedał na kasie | CREATE FUNCTION pracownik\_miesiaca(@data DATE)  RETURNS table  AS  RETURN  (SELECT TOP 1 p.id\_pracownik, SUM(pr.cena\_za\_porcje \* z.ilosc) AS Suma FROM Pracownicy p  JOIN Kasa k ON p.id\_pracownik = k.id\_pracownik  JOIN Zamowienia z ON k.id\_kasa = z.id\_kasa  JOIN Produkty pr ON z.id\_produkt = pr.id\_produkt  where month(z.data\_zamowienia) = month(@data)  GROUP BY p.id\_pracownik  ORDER BY Suma DESC) |
| 16 | Funkcja sprawdzająca ile kosztowali artyści w danym okresie czasu. | CREATE FUNCTION Ile\_kosztowali\_artysci (@odKiedy DATE, @doKiedy DATE)  RETURNS FLOAT  AS  BEGIN  RETURN (SELECT SUM(A.placa) FROM Artysci A  JOIN Impreza\_artysci IA ON A.id\_artysta = IA.id\_artysta  JOIN Impreza I ON IA.id\_impreza = I.id\_impreza  WHERE I.data\_imprezy BETWEEN @odKiedy AND @doKiedy)  END |
| 17 | Funkcja sprawdzająca która usługa sprzedała się najlepiej w danym okresie czasu. | CREATE FUNCTION najlepsza\_usluga(@od date, @do date)  RETURNS VARCHAR(20)  BEGIN  RETURN  (SELECT rodzaj\_uslugi FROM  (SELECT top 1 US.rodzaj\_uslugi, count(ZA.id\_zamowienie) AS ilosc FROM Usluga AS US  JOIN StanowiskaUslug AS SUS ON US.id\_usluga = SUS.id\_usluga  JOIN Zamowienia AS ZA ON SUS.id\_stanowisko = ZA.id\_stanowisko  WHERE ZA.data\_zamowienia >= @OD AND ZA.data\_zamowienia <= @DO  GROUP BY US.rodzaj\_uslugi  order by ilosc desc  ) AS X  )  END |
| 18 | Funkcja wypisująca pracowników z podanego miasta | CREATE FUNCTION pracownicy\_z\_miasta (@miasto VARCHAR(20))  RETURNS TABLE  AS  RETURN (  SELECT imie,nazwisko,nazwa\_stanowiska,stawka\_godzinowa FROM Pracownicy P JOIN StanowiskaPracy SP ON P.id\_stanowisko=SP.id\_stanowisko  WHERE miasto = @miasto  ) |
| 19 | Funkcja wypisująca najlepiej sprzedający się produkt w danym miesiącu. | CREATE FUNCTION najlepszy\_produkt(@miesiac date)  returns varchar(20)  BEGIN  RETURN (SELECT nazwa\_produktu FROM (SELECT TOP 1 P.nazwa\_produktu, SUM(Z.ilosc \* P.cena\_za\_porcje) AS obrut FROM Produkty AS P  JOIN Zamowienia AS Z ON P.id\_produkt = Z.id\_produkt  WHERE MONTH(Z.data\_zamowienia) = MONTH(@miesiac)  GROUP BY P.nazwa\_produktu  ORDER BY obrut DESC  ) AS X  )  END |
| 20 | Funkcja wypisująca ile dana kategoria produktów zarobiła do tej pory. | CREATE FUNCTION zarobki\_kategorii(@id\_kategoria INT)  RETURNS FLOAT  AS  BEGIN  RETURN (SELECT SUM(Z.ilosc \* P.cena\_za\_porcje) FROM Produkty P JOIN Zamowienia Z  ON P.id\_produkt = Z.id\_produkt WHERE P.id\_kategoria = @id\_kategoria)  END |

# Użyte Procedury w Projekcie:

|  |  |  |
| --- | --- | --- |
| Nr. | Opis | Kod |
| 1 | Procedura pozwalająca sprzedać towar. | CREATE PROC sprzedaj\_towar  @id\_towar INT,  @ilosc INT,  @kasa INT  AS  IF @ilosc < 0  BEGIN  RAISERROR('Nie moze sprzedac ujemnej ilosci towaru',1,1)  END  INSERT INTO Zamowienia(ilosc, id\_produkt, id\_kasa, data\_zamowienia)  VALUES (@ilosc, @id\_towar, @kasa, getdate())  GO |
| 2 | Procedura zamawiania towaru od dostawców. | CREATE PROCEDURE Zamowienie\_towaru(@id\_produkt INT ,@ilosc FLOAT)  AS  IF @ilosc < 0  BEGIN  RAISERROR('Nie moze zamowic ujemnej ilosci towaru',1,1)  END  ELSE  BEGIN  UPDATE Produkty  SET [stan\_magazyn(w kg)] = [stan\_magazyn(w kg)] + @ilosc  WHERE id\_produkt = @id\_produkt  END  GO  ---tu dodamy 2,5 litra zubrowki do magazynu ---  EXEC Zamowienie\_towaru 1, -2.5  GO |
| 3 | Procedura przyznająca daną premie danym pracownikom. | CREATE PROC przyznaj\_premie  @id\_pracownik INT,  @id\_premii INT  AS  UPDATE Wyplata  SET id\_premia = @id\_premii  WHERE id\_pracownik = @id\_pracownik AND MONTH(data\_wyplaty) = MONTH(GETDATE())  GO |
| 4 | Procedura dodania urlopu na określoną ilość dni danemu pracownikowi | CREATE PROCEDURE Dodaj\_urlop (@id\_pracownik INT, @na\_ile\_dni INT)  AS  IF @na\_ile\_dni < 1  RAISERROR('Nie mozna dac urlopu na 0 lub mniej dni',1,1)  ELSE  BEGIN  DECLARE @doKiedy DATE  SET @doKiedy = (SELECT GETDATE() + @na\_ile\_dni)  INSERT INTO Urlopy VALUES (@id\_pracownik,CAST(GETDATE() AS DATE),@doKiedy)  END |
| 5 | Procedura sprzedaży usługi. Jeśli stanowisko jest zajęte, nie da się sprzedać i wyskakuje komunikat. | CREATE PROCEDURE sprzedaj\_usluge  @id\_stanowisko INT,  @ilosc INT,  @kasa INT  AS  IF @ilosc < 0  RAISERROR('Nie moze zamowic ujemnej ilosci uslugi',1,1)  ELSE  BEGIN  IF EXISTS (SELECT czy\_wolne FROM StanowiskaUslug WHERE id\_stanowisko=@id\_stanowisko AND czy\_wolne = 1 )  BEGIN  print ('Jest wolne stanowisko')  INSERT INTO Zamowienia(ilosc, id\_stanowisko, id\_kasa, data\_zamowienia)  VALUES (@ilosc, @id\_stanowisko, @kasa, getdate())  END  ELSE  print ('stanowisko jest zajęte')  END |
| 6 | Procedura zwalnia dane stanowisko usługi jeśli jest ono zajęte. | CREATE PROCEDURE ZwolnijStanowisko (@nrStanowiska INT)  AS  IF (SELECT czy\_wolne FROM StanowiskaUslug WHERE id\_stanowisko = @nrStanowiska) = 0  BEGIN  UPDATE StanowiskaUslug  SET czy\_wolne = 1  WHERE id\_stanowisko = @nrStanowiska  END  ELSE  PRINT 'Stanowisko jest wolne!!!' |
| 7 | Procedura dodawania pracownika, błąd gdy chcemy dodać szefa (wtedy musi dodać administrator) | CREATE PROC zatrudnienie\_pracownika  @imie varchar(20),  @nazwisko varchar(20),  @ulica varchar(40),  @nr\_domu int,  @misato varchar(30),  @wojewodztwo varchar(30),  @kod\_pocztowy int,  @obywatelstwo varchar(20),  @id\_stanowisko INT,  @data\_zatrudnienia date  AS  IF @id\_stanowisko = 11  PRINT 'Nie mozna dodac szefa'  ELSE  INSERT INTO Pracownicy VALUES  (@imie, @nazwisko, @ulica, @nr\_domu, @misato,  @wojewodztwo, @kod\_pocztowy, @obywatelstwo, @id\_stanowisko, @data\_zatrudnienia) |
| 8 | Procedura zmiany artysty z danej imprezy na nowo dodanego artystę. | CREATE PROCEDURE ZmienArtyste (@nazwa VARCHAR(20),@kontakt INT, @profesja VARCHAR(20), @placa MONEY, @id\_impreza INT, @id\_artysta INT)  AS  INSERT INTO Artysci VALUES (@nazwa,@kontakt,@profesja,@placa)  UPDATE Impreza\_artysci  SET id\_artysta = (SELECT TOP 1 id\_artysta FROM Artysci ORDER BY id\_artysta DESC)  WHERE id\_impreza = @id\_impreza AND id\_artysta = @id\_artysta  --przyklad--  EXEC ZmienArtyste Marianek,664855555,Piosenkarz,3000,2,4  SELECT \* FROM Artysci JOIN Impreza\_artysci ON Artysci.id\_artysta=Impreza\_artysci.id\_artysta |
| 9 | Procedura zwalniania pracownika. | CREATE PROC zwolnij\_pracownika  @id\_pracownik INT  AS  IF EXISTS (SELECT id\_pracownik FROM Pracownicy WHERE id\_pracownik = @id\_pracownik)  DELETE Pracownicy  WHERE id\_pracownik = @id\_pracownik  ELSE  PRINT('Nie ma takiego pracownika') |
| 10 | Procedura zwiększająca zarobki danej kategorii pracowników. | CREATE PROCEDURE zwieksz\_zarobki (@procent FLOAT,@id\_stanowisko INT)  AS  UPDATE StanowiskaPracy  SET stawka\_godzinowa = stawka\_godzinowa + (stawka\_godzinowa \* @procent)/100  WHERE id\_stanowisko = @id\_stanowisko |
| 11 | Procedura dodania pracownika o najmniejszej ilości godzin w danym miesiącu do następnej imprezy. | CREATE PROCEDURE DodajPracownika (@data\_wyplaty DATE)  AS  IF (SELECT TOP 1 data\_imprezy FROM Impreza Order by id\_impreza DESC ) > GETDATE()  BEGIN  DECLARE @id\_pracownik INT  SET @id\_pracownik = (SELECT TOP 1 id\_pracownik FROM Wyplata WHERE data\_wyplaty = @data\_wyplaty ORDER BY ilosc\_godzin )  DECLARE @id\_impreza INT  SET @id\_impreza = (SELECT TOP 1 id\_impreza FROM Impreza ORDER BY id\_impreza DESC)  IF NOT EXISTS (SELECT id\_pracownik FROM Impreza\_pracownicy WHERE id\_pracownik = @id\_pracownik AND id\_impreza = @id\_impreza)  INSERT INTO Impreza\_pracownicy VALUES (@id\_impreza,@id\_pracownik)  ELSE  PRINT 'Pracownik jest juz na imprezie'  END  ELSE  PRINT 'Nie ma nastepnej imprezy w bazie' |

# Użyte Wyzwalacze w Projekcie:

|  |  |  |
| --- | --- | --- |
| Nr. | Opis | Kod |
| 1 | Wyzwalacz który aktualizuje stan kasy zaraz po zamówieniu. | CREATE TRIGGER stan\_Kasy on Zamowienia  AFTER INSERT  AS  BEGIN  IF (SELECT id\_produkt FROM inserted) IS NOT NULL  BEGIN  DECLARE @nowe table (ilosc int, id\_kasa int, cena money)  insert into @nowe SELECT ins.ilosc, ins.id\_kasa, pro.cena\_za\_porcje  FROM inserted as ins join Produkty as pro on ins.id\_produkt = pro.id\_produkt  UPDATE Kasa SET stan+=(select ilosc\*cena from @nowe)  WHERE id\_kasa = (SELECT id\_kasa FROM @nowe)  END  IF (SELECT id\_stanowisko FROM inserted) IS NOT NULL  BEGIN  DECLARE @nowe1 table (ilosc int, id\_kasa int, cena money)  insert into @nowe1 SELECT ins.ilosc, ins.id\_kasa, usl.cena  FROM inserted as ins  join StanowiskaUslug as sta on ins.id\_stanowisko = sta.id\_stanowisko  join Usluga as usl on sta.id\_usluga = usl.id\_usluga  UPDATE Kasa SET stan+=(select ilosc\*cena from @nowe1)  WHERE id\_kasa = (SELECT id\_kasa FROM @nowe1)  END  END  GO |
| 2 | Wyzwalacz aktualizujący stan magazynu po zamówieniu. | CREATE TRIGGER stan\_magazynu ON Zamowienia  AFTER INSERT  AS  BEGIN  IF (SELECT id\_produkt FROM inserted) IS NOT NULL  BEGIN  DECLARE @Ilosc FLOAT  SET @Ilosc = (SELECT ilosc FROM inserted) \* (SELECT gramatura\_porcji FROM Produkty P JOIN inserted I ON P.id\_produkt=I.id\_produkt )/1000  UPDATE Produkty  SET [stan\_magazyn(w kg)] = [stan\_magazyn(w kg)] - @Ilosc  WHERE id\_produkt = (SELECT id\_produkt FROM inserted)  PRINT 'Produkt sprzedano poprawnie'  END  END  GO |
| 3 | Wyzwalacz który zmienia stan stanowiska usługi po zamówieniu na ‘zajęte’ . | CREATE TRIGGER zajete ON Zamowienia  AFTER INSERT  AS  BEGIN  IF (SELECT id\_stanowisko FROM inserted) IS NOT NULL  BEGIN  IF (SELECT S.czy\_wolne FROM StanowiskaUslug S join inserted I ON S.id\_stanowisko=I.id\_stanowisko) = 1  BEGIN  UPDATE StanowiskaUslug  SET czy\_wolne = 0  WHERE id\_stanowisko = (SELECT id\_stanowisko FROM inserted)  PRINT 'Stanowisko zostalo zajete'  END  ELSE  PRINT 'Stanowisko jest juz zajete'  END  END  GO |
| 4 | Wyzwalacz który po usunięciu zamówienia  powraca do stanu pierwotnego stan magazynu. | CREATE TRIGGER powrot\_do\_stanu\_pierwotnego ON Zamowienia  AFTER DELETE  AS  BEGIN  IF (SELECT id\_produkt FROM deleted) IS NOT NULL  BEGIN  DECLARE @nowe table (ilosc int, id\_kasa int, cena money)  insert into @nowe SELECT del.ilosc, del.id\_kasa, pro.cena\_za\_porcje  FROM deleted as del join Produkty as pro on del.id\_produkt = pro.id\_produkt  UPDATE Kasa SET stan-=(select ilosc\*cena from @nowe)  WHERE id\_kasa = (SELECT id\_kasa FROM @nowe)  DECLARE @Ilosc FLOAT  SET @Ilosc = (SELECT ilosc FROM deleted) \* (SELECT gramatura\_porcji FROM Produkty as P JOIN deleted as del ON P.id\_produkt = del.id\_produkt )/1000  UPDATE Produkty  SET [stan\_magazyn(w kg)] += @Ilosc  WHERE id\_produkt = (SELECT id\_produkt FROM deleted)  END  IF (SELECT id\_stanowisko FROM deleted) IS NOT NULL  BEGIN  DECLARE @nowe1 table (ilosc int, id\_kasa int, cena money)  insert into @nowe1 SELECT del.ilosc, del.id\_kasa, usl.cena  FROM deleted as del  join StanowiskaUslug as sta on del.id\_stanowisko = sta.id\_stanowisko  join Usluga as usl on sta.id\_usluga = usl.id\_usluga  UPDATE Kasa SET stan-=(select ilosc\*cena from @nowe1)  WHERE id\_kasa = (SELECT id\_kasa FROM @nowe1)  END  END  GO |
| 5 | Wyzwalacz informuje po usunięciu pracownika czy jest jeszcze ktoś na to stanowisko. | CREATE TRIGGER czy\_jest\_inny\_pracownik ON Pracownicy  AFTER DELETE  AS  BEGIN  DECLARE @stanowisko INT  SET @stanowisko = (SELECT id\_stanowisko FROM deleted)  IF EXISTS (SELECT P.id\_pracownik FROM Pracownicy P LEFT JOIN deleted D  ON P.id\_pracownik = D.id\_pracownik WHERE P.id\_stanowisko = @stanowisko AND D.id\_pracownik IS NULL)  PRINT 'Usunieto pracownika ! Sa jeszcze pracownincy na to stanowisko'  ELSE  PRINT 'SZUKAJ NOWEGO PRACOWNIKA'  END  GO |
| 6 | Przyznaje wszystkim premie świąteczną w grudniu. | CREATE TRIGGER premia\_swiateczna ON wyplata  AFTER INSERT  AS  IF DATEPART(MONTH, (SELECT data\_wyplaty FROM inserted)) = 12  BEGIN  UPDATE Wyplata  SET id\_premia = 3 --Premia świąteczna  WHERE MONTH(data\_wyplaty) = 12  END  GO |
| 7 | Gdy brakuje produktu w magazynie, pokazuje jego aktualny stan przy sprzedaży tego produktu. | CREATE TRIGGER gdy\_malo ON Zamowienia  AFTER INSERT  AS  BEGIN  IF (SELECT id\_produkt from inserted) IS NOT NULL  BEGIN  DECLARE @id\_produktu INT  DECLARE @stan\_magazyn FLOAT  SET @id\_produktu = (SELECT id\_produkt FROM inserted)  SET @stan\_magazyn = (SELECT [stan\_magazyn(w kg)] FROM Produkty WHERE id\_produkt = @id\_produktu)  IF @stan\_magazyn < 5  PRINT 'DOMÓW PRODUKT ! JEST GO TYLKO' + CAST(@stan\_magazyn AS VARCHAR(50)) + ' kg'  END  END  INSERT INTO Zamowienia VALUES (3,25,NULL,1,GETDATE())  GO |
| 8 | Wyzwalacz usuwa zamówienia starsze niż 5 lat. | CREATE TRIGGER czyszczenie\_pamieci ON Zamowienia  AFTER INSERT  AS  WHILE (SELECT MIN(data\_zamowienia) FROM Zamowienia) < GETDATE()-YEAR(5)  BEGIN  DELETE Zamowienia  WHERE data\_zamowienia = (SELECT MIN(DATA\_ZAMOWIENIA) FROM ZAMOWIENIA WHERE data\_zamowienia < GETDATE()-YEAR(5))  PRINT 'Z TABELI ZAMOWIENIA ZOSTAŁ USUNIĘTY REKORD STARSZY NIŻ 5 LAT'  END  GO |
| 9 | Wyzwalacz włącza kasę po zamówieniu z danej kasy, jeśli była wcześniej wyłączona. | CREATE TRIGGER wlacz\_kase ON Zamowienia  AFTER INSERT  AS  BEGIN  DECLARE @id\_kasa TINYINT  SET @id\_kasa = (SELECT id\_kasa FROM inserted)  IF (SELECT czy\_aktywna FROM Kasa WHERE id\_kasa = @id\_kasa) = 0  BEGIN  UPDATE Kasa  SET czy\_aktywna = 1  WHERE id\_kasa = @id\_kasa  PRINT 'Kasa zostala uruchomiona'  END  END  INSERT INTO Zamowienia VALUES (3,9,NULL,3,GETDATE())  GO |
| 10 | Wyzwalacz przelewa 200 zł z kasy w której jest najwięcej pieniędzy do nowo powstałej kasy. | CREATE TRIGGER dodaj\_kase ON Kasa  AFTER INSERT  AS  BEGIN  DECLARE @id\_kasa TINYINT  SET @id\_kasa = (SELECT TOP 1 id\_kasa FROM Kasa ORDER BY stan)  IF (SELECT stan FROM Kasa WHERE id\_kasa = @id\_kasa) >= 300  BEGIN  UPDATE Kasa  SET stan += 200  WHERE id\_kasa = (SELECT id\_kasa FROM inserted)  UPDATE Kasa  SET stan -=200  WHERE id\_kasa = @id\_kasa  END  ELSE  BEGIN  PRINT 'W zadnej kasie nie ma wiecej niz 300zl '  ROLLBACK  END  END  GO |

# Użyte Widoki w Projekcie:

|  |  |  |
| --- | --- | --- |
| Nr. | Opis | Kod |
| 1 | Wypisuje telefony i nazwy artystów którzy jeszcze nie byli na imprezie w klubie. | CREATE VIEW nie\_byli\_na\_imprezie  AS  SELECT A.nazwa, A.kontakt FROM Artysci AS A FULL JOIN Impreza\_artysci AS IA ON A.id\_artysta = IA.id\_artysta  WHERE IA.id\_impreza IS NULL |
| 2 | Wypisuje ilość sprzedanych produktów w ostatnim miesiącu. | CREATE VIEW sprzedane\_produkty\_w\_ostatnim\_msc  AS  SELECT Z.id\_produkt,P.nazwa\_produktu, SUM(ilosc) ilosc\_sprzedanych  FROM Produkty P JOIN Zamowienia Z ON P.id\_produkt=Z.id\_produkt  WHERE Z.data\_zamowienia BETWEEN CAST(GETDATE()-30 AS DATE) AND CAST(GETDATE() AS DATE)  GROUP BY Z.id\_produkt,P.nazwa\_produktu |