

PROJEKT ZALICZENIOWY

BAZY DANYCH

JAKUB LEMKA; Analityka Gospodarcza gr. 2.2.

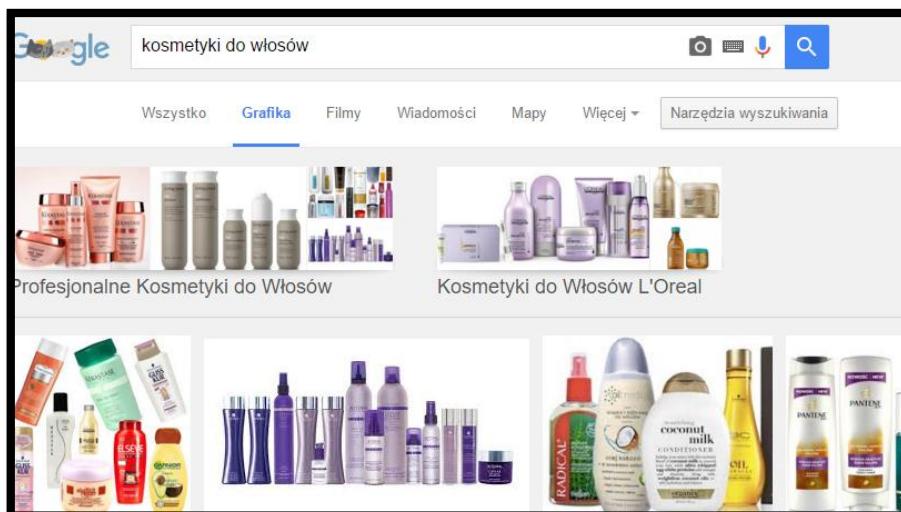
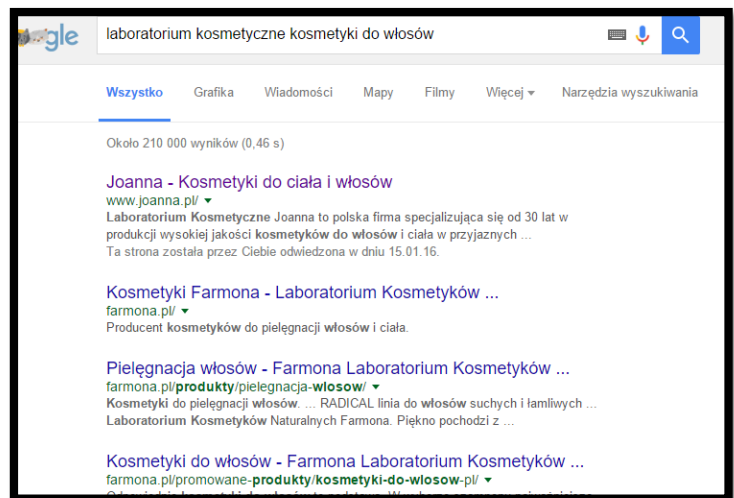
TEMAT NR 52:
LABORATORIUM KOSMETYCZNE
KOSMETYKI DO WŁOSÓW

Spis treści

1.	Wprowadzenie.....	2
2.	Założenia, wymagania, struktura	3
3.	Tabele	4
3.1.	Adresy.....	4
3.2.	Kategoria.....	4
3.3.	Klienci.....	4
3.4.	Pozycja_zamówienie.....	5
3.5.	Pracownicy	5
3.6.	Proces	5
3.7.	Produkty.....	6
3.8.	Stanowisko	6
3.9.	Zamówienia.....	6
4.	Diagram	7
5.	Wypełnienie tabel, dodawanie rekordów	7
6.	Widoki	9
6.1.	Najlepsza_sprzedaż_pracowników.....	9
6.2.	Stali_klienci	9
6.3.	Pracownik_na_minus.....	10
6.4.	Szczegóły zamówień	11
7.	Funkcje	12
7.1.	Funkcje tabelowe	12
7.1.1.	Klient_info	12
7.1.2.	Pracownik_info.....	12
7.1.3.	KategoriaProduktu.....	13
7.1.4.	Szczegóły_danego_zamówienia	14
7.2.	Funkcje skalarne	15
7.2.1.	Staż_pracy	15
7.2.2.	Ilość_zamówień	15
8.	Triggery.....	16
8.1.	BLOKADA_NIEGOTOWYCH.....	16
8.2.	Uprawnienia.....	17
9.	Inne elementy.....	18
10.	Podsumowanie.....	18

1. Wprowadzenie.

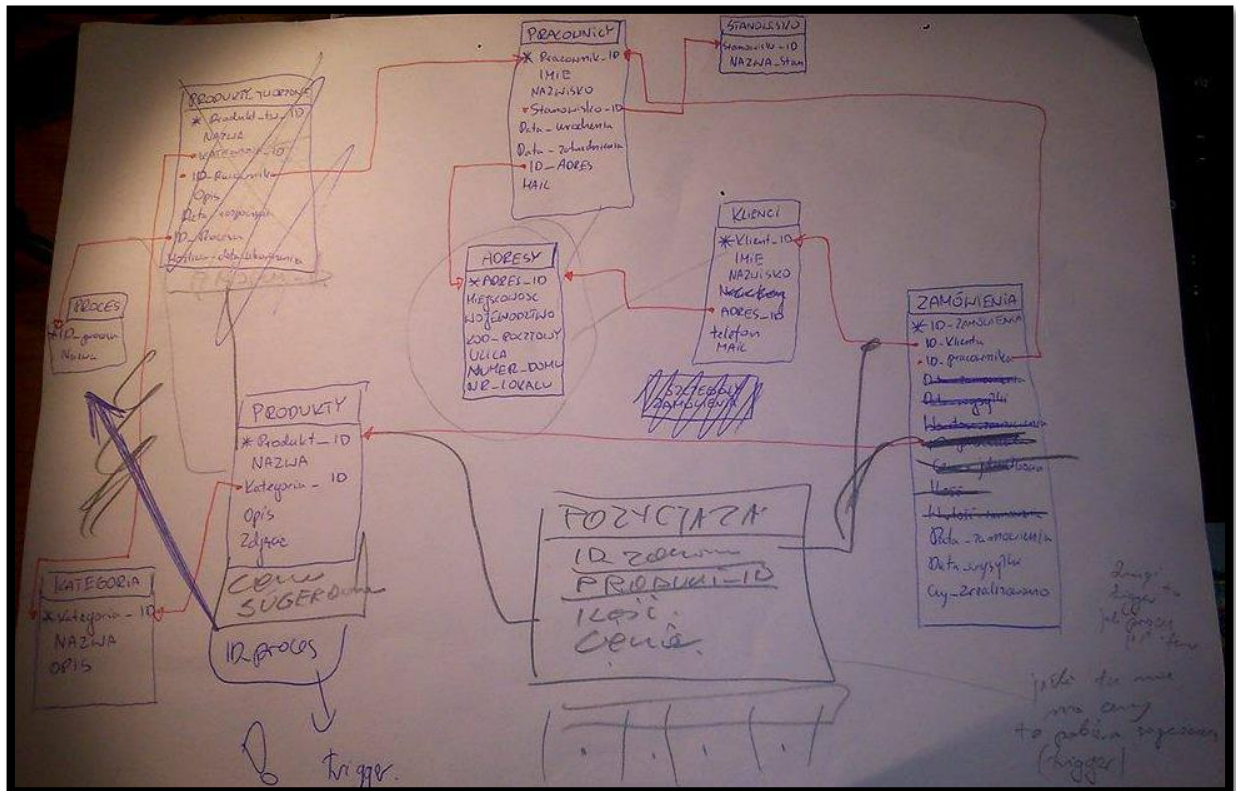
Na początku grudnia 2015 roku w drodze losowania tematem mojego projektu zaliczeniowego stało się laboratorium kosmetyczne – kosmetyki do włosów. Już, gdy losowanie się dokonało wiedziałem, iż pierwszą przeszkodą będzie zrozumienie, czym właściwie zajmuje się takie laboratorium, oraz jakie są kosmetyki do włosów poza szamponem. Z pomocą przyszła niezawodna (zazwyczaj) wyszukiwarka Google, oraz niezwykle pomocne przedstawicielki płci pięknej z grona znajomych.



Po dokonaniu małego 'researchu' doszedłem do wniosku, iż takie laboratorium to nijako hurtownia bądź sklep internetowy, który swą działalność w sieci nazywa laboratorium. Prościej ujmując, jest to hurtownia zazwyczaj hurtownia internetowa, sprzedająca produkty swojej marki.

2. Założenia, wymagania, struktura

Po zgłębieniu tematu jakim przyjdzie mi przez kolejne dni się zajmować przyszedł czas na zaprojektowanie mojej bazy danych. Zacząłem od wstępnego rozrysowania diagramu całej bazy. Jak się później okazało owa kartka, pomimo kilkukrotnego zmieniania rzeczy ważnych jak i małych szczegółów, towarzyszyła mi przez cały czas pracy.



Przed wszystkim bazę oparłem na paru bazach stworzonymi na potrzeby wszelkich sklepów czy hurtowni. By jednak podkreślić i zawrzeć coś, co by nakierowało innych na to, iż jest to baza danych dla laboratorium dodałem tabelę dotyczącą w jakiej fazie jest dany produkt (Proces). Może on być gotowy, w fazie tworzenia bądź w fazie dokonywania testów kosmetyków. W sumie stworzyłem 9 tabel. Jedna z nich – Pozycja_zamówienie posiada podwójny klucz główny. Jest ona stworzona po to, by możliwe było zamówienie kilku produktów.

9 stworzonych na potrzeby bazy danych dla laboratorium kosmetycznego to: (tabele opisane po kolei w kolejności alfabetycznej):

3. Tabele

3.1. Adresy

Tabela ta przechowuje adresy zarówno klientów jak i pracowników. Sprawdza się idealnie w momencie, gdy pracownik zostaje klientem – dokonuje zakupu w naszym sklepie. Ponadto pozwala zachować większą przejrzystość w tabelach, z którymi jest połączona relacją: pracownicy i klienci. Zawiera informacje o miejscowości, województwie zamieszkania danej osoby, jego dokładnym adresie. Dzięki temu w tabelach Pracownicy oraz Klienci mamy mniej pól co znacznie ułatwia szukanie najważniejszych informacji (nazwisko, mail czy telefon). Adres_ID w tabeli Adresy jest kluczem główny – klicze obce o takiej samej nazwie w obu wyżej wymienionych tabelach.

	Column Name	Data Type	Allow Nulls
▶🔑	Adres_ID	int	<input type="checkbox"/>
	Miejscowość	nvarchar(50)	<input type="checkbox"/>
	Województwo	nvarchar(50)	<input type="checkbox"/>
	Kod_Pocztowy	nvarchar(10)	<input type="checkbox"/>
	Ulica	nvarchar(50)	<input type="checkbox"/>
	Numer_domu	varchar(10)	<input type="checkbox"/>
	Numer_lokalu	varchar(10)	<input checked="" type="checkbox"/>

3.2. Kategoria

Tabela Kategoria jest prostą tabelą, gdzie klucz główny Kategoria_ID jest połączony relacją z kluczem obcym w tabeli Produkty.

	Column Name	Data Type	Allow Nulls
▶🔑	Kategoria_ID	int	<input type="checkbox"/>
	Nazwa_kategorii	nvarchar(50)	<input type="checkbox"/>
	Opis	nvarchar(100)	<input checked="" type="checkbox"/>

3.3. Klienci

Tabela Klienci zawiera informacje o wszystkich klientach laboratorium. Oprócz imienia i nazwiska odnajdziemy tu jego/jej telefon oraz mail, a także ID adresu, które to pole jest kluczem obcym i nawiązuje relację z wcześniej opisaną tabelą z adresami. Ważną informacją jest tu fakt, iż wszystkie pola muszą być wypełnione, zarówno mail jak i telefon przy dodawaniu danych nie może pozostać 'NULL'.

	Column Name	Data Type	Allow Nulls
▶🔑	Klient_ID	int	<input type="checkbox"/>
	Imię	nvarchar(50)	<input type="checkbox"/>
	Nazwisko	nvarchar(50)	<input type="checkbox"/>
	ID_Adres	int	<input type="checkbox"/>
	Telefon	nvarchar(50)	<input type="checkbox"/>
	Mail	nvarchar(50)	<input type="checkbox"/>

3.4. Pozycja_zamówienie

Tabela ta powstała po to, by możliwe było zamówienie więcej niż jednego produktu. Posiada podwójny klucz główny, informację o cenie (cena zawsze jest uzgadniana indywidualnie z klientem, jednak w razie potrzeby w tabeli Produkty jest podana cena sugerowana) oraz o ilości w jakiej klient zamówił dany produkt. Wartość zamówienia to iloczyn ilości oraz ceny.

```
CREATE TABLE [dbo].[Pozycja_zamówienie](
    [ID_zamówienie] [int] NOT NULL,
    [ID_produkt] [int] NOT NULL,
    [Ilość] [int] NOT NULL,
    [Cena] [money] NOT NULL,
    [Wartość_zamówienia] AS ([Ilość]*[Cena]),
    CONSTRAINT [PK_Pozycja_zamówienie] PRIMARY KEY CLUSTERED
    (
        [ID_zamówienie] ASC,
        [ID_produkt] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
    ) ON [PRIMARY]
```

3.5. Pracownicy

Tabela pracownicy strukturą przypomina bardzo tabelę klientów. ID pracownika zostaje uzupełnione automatycznie przy dodawaniu nowego członka zakładu (IDENTITY). Również posiadamy tu informacje o imieniu, nazwisku, mailu i telefonie danego osobnika, jak i ID jego adresu zamieszkania. Dodatkowo w tabeli jest pole Stanowisko_ID – klucz obcy relacji z tabelą stanowisko. Dodatkowo mamy dwa ostatnie pola Data_urodzenia i Data_zatrudnienia. To długie jest o tyle ważne, że dzięki niemu i łatwo sprawdzimy (jak – opisane zostanie w dalszej części raportu) staż pracy każdego pracownika.

```
SQLQuery1.sql - US...omputer\user (52))* X
use BD_Lemka_projekt
CREATE TABLE dbo.Pracownicy
(Pracownik_ID INT PRIMARY KEY IDENTITY(1,1) ,
Imię NVARCHAR(50) NOT NULL,
Nazwisko NVARCHAR(50) NOT NULL,
Stanowisko_ID INT NOT NULL,
Data_urodzenia DATE NOT NULL,
Data_zatrudnienia DATE NOT NULL,
ID_Adres INT NOT NULL,
Mail nvarchar(50) NOT NULL,
)
```

3.6. Proces

Tabela proces była największą wątpliwością przy projektowaniu opisywanej bazy danych. Po kilku konsultacjach z Nauczycielem Prowadzącym postanowiłem jednak z niej nie rezygnować. Mimo tego, co

przeczytałem w internecie – słowo „laboratorium” zobowiązuje. Tabela ta posiada jedynie dwa pola – jedno, klucz główny to ID procesu a drugie, co oczywiste, nazwa. Jest to o tyle istotne, iż niegotowych produktów nie można sprzedawać, i takie ograniczenie zostanie później wprowadzone.

	Column Name	Data Type	Allow Nulls
🔍	ID_Procesu	int	<input type="checkbox"/>
	Nazwa_procesu	nvarchar(50)	<input type="checkbox"/>

3.7. Produkty

Można by rzec, rdzeń naszej bazy danych. Bez produktów laboratorium, sklep, hurtownia nie miałyby racji bytu. Każdy produkt posiada swój ID = klucz główny, nazwę, ID kategorii, ID procesu, najczęściej jest opisany (jednak w tym polu możliwe jest pozostawienie go pustym) oraz zawsze posiada też sugerowaną cenę.

	Column Name	Data Type	Allow Nulls
🔑	Produkt_ID	int	<input type="checkbox"/>
	Nazwa_produkту	nvarchar(50)	<input type="checkbox"/>
	ID_kategoria	int	<input type="checkbox"/>
	ID_proces	int	<input type="checkbox"/>
	Opis	nvarchar(100)	<input checked="" type="checkbox"/>
	Cena_sugerowana	money	<input type="checkbox"/>

3.8. Stanowisko

Kolejna, po Proces, tabela posiadająca dwa pola. Jak sama nazwa wskazuje, zawiera nazwy stanowisk pracowników laboratorium.

	Column Name	Data Type	Allow Nulls
🔑	Stanowisko_ID	int	<input type="checkbox"/>
	Nazwa_stanowiska	nvarchar(50)	<input type="checkbox"/>

3.9. Zamówienia

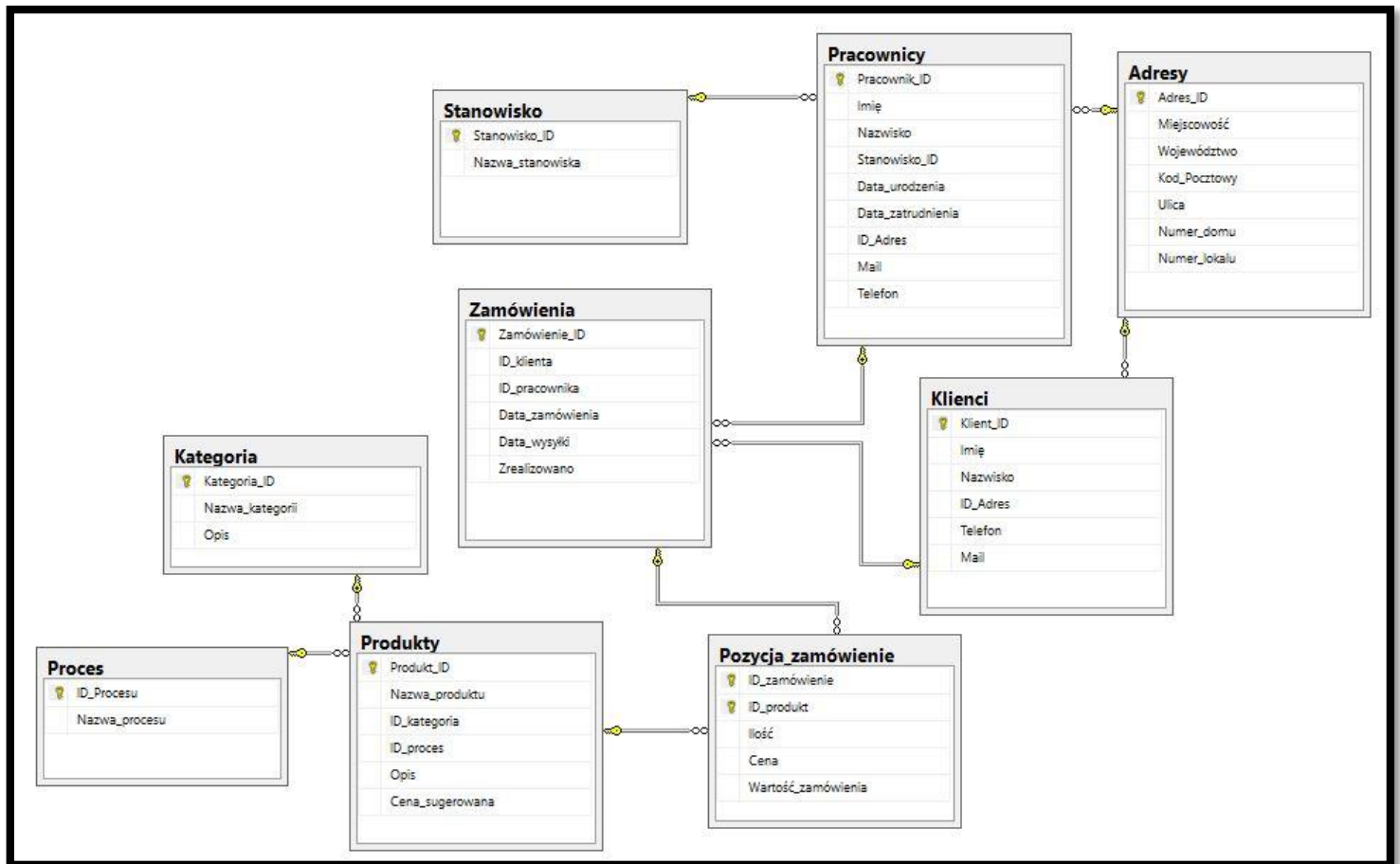
Ostatnia opisywana tabela absolutnie nie jest najmniej ważną. Każde zamówienie posiada swoje ID, ID klienta składającego zamówienia oraz pracownika odpowiedzialnego za dane zamówienie. Posiada też pola z datą zamówienia oraz datą wysyłki. Data wysyłki jest wprowadzana PO wysłaniu zamówienia. Jak 'jest NULL' to zamówienie wciąż czeka na wysłanie. By było to bardziej przejrzyste

zostało dodane osobne pole – Zrealizowano. Gdy wspomniana data wysyłki jest pusta, wtedy pojawia się słowo 'NIE', gdy data wysyłki jest wprowadzona automatycznie zmienia się na 'TAK'.

	Column Name	Data Type	Allow Nulls
🔑	Zamówienie_ID	int	<input type="checkbox"/>
	ID_klienta	int	<input type="checkbox"/>
	ID_pracownika	int	<input type="checkbox"/>
	Data_zamówienia	date	<input type="checkbox"/>
	Data_wysyłki	date	<input checked="" type="checkbox"/>
	Zrealizowano		<input type="checkbox"/>

4. Diagram

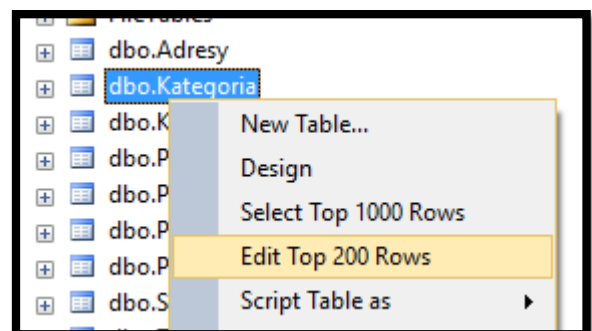
Opisane wyżej 9 tabel połączonych relacjami wraz z zaznaczonymi kluczami głównymi przedstawia DIAGRAM



5. Wypełnienie tabel, dodawanie rekordów

Po zaprojektowaniu, i przeniesieniu projektu do 'rzeczywistości' jaką w tym przypadku jest SQL Server Management Studio przyszedł czas na wypełnienie rekordów. W większości przypadków by to uczynić używałem edytora zamiast komend, gdyż tak szło to znacznie szybciej.

Jedyna zmiana nastąpiła podczas wypełniania jednej z tabel a konkretnie tabeli Pozycja_zamówienie. To tu sporym utrudnieniem, co zabierało mnóstwo czasu był fakt, że pola muszą być wypełniane jedno za drugim dla każdego wiersza. Nie mogłem dodać najpierw np. 20 rekordów jedynie do pola ID_zamówienie, gdyż zabroniłem wcześniej pozostawienie innych pól pustych. To nijako zmusiło mnie do wprowadzenia danych w programie Excel, a następnie ich import do SQL'a.



	A	B	C	D	E	F	G
1	ID_zamówienie	ID_produkt	Ilość	Cena	Wartość_zamówienia		
2		1	1	30	20		
3		1	9	30	45		
4		2	26	10	88		
5		3	29	10	30		
6		3	1	35	20		
7		3	3	85	20		
8		3	21	15	44		
9		3	22	15	47		
10		4	19	100	15,5		
11		4	20	100	38		
12		5	12	50	31,1		
13		6	1	100	18		
14		7	23	44	33		
15		8	9	20	45		
16		8	23	60	32		
17		8					
18		8					
19		8					
20		9					
21		10					

SQL Server Import and Export Wizard

Choose a Data Source
Select the source from which to copy data.

Data source: Microsoft Excel

Excel connection settings

Excel file path:
 Browse...

Excel version:
Microsoft Excel 97-2003

☒ First row has column names

Select Source Tables and Views

Choose one or more tables and views to copy.

Tables and views:

<input checked="" type="checkbox"/>	Source: C:\Users\user\Desktop\DANE1.xls	Destination: USER-KOMPUTER\SQLENT201...
<input checked="" type="checkbox"/>	'Arkusz1\$'	[dbo].[Pozycja_zamówienie]

6. Widoki

6.1. Najlepsza_sprzedaż_pracowników

Pierwszy wykonany w bazie widok przedstawia pracowników, którzy zajęli się największą liczbą zamówień. By do takich pracowników mieć od razu kontakt podany jest jego mail. Pole PRACOWNIK to imię+nazwisko pracownika. Taki widok pozwala sprawdzić który z pracowników opiekuje się największą liczbą zamówień.

```
CREATE VIEW Najlepsza_sprzedaż_pracowników AS
SELECT Pracownicy.Pracownik_ID, Pracownicy.Imię + ' ' + Pracownicy.Nazwisko AS PRACOWNIK,
Pracownicy.Mail AS Mail_pracownika, COUNT(Zamówienia.Data_zamówienia) AS Ilość_zrealizowanych_zamówień
FROM Zamówienia INNER JOIN
      Pracownicy ON Zamówienia.ID_pracownika = Pracownicy.Pracownik_ID
GROUP BY Pracownicy.Pracownik_ID, Pracownicy.Imię + ' ' + Pracownicy.Nazwisko, Pracownicy.Mail

select * from Najlepsza_sprzedaż_pracowników
order by Ilość_zrealizowanych_zamówień desc
```

	Pracownik_ID	PRACOWNIK	Mail_pracownika	Ilość_zrealizowanych_zamówień
1	2	Michalina Pelka	pelka_m@wp.pl	8
2	5	Katarzyna Miotke	miotke_kaska@yahoo.com	8
3	3	Marta Kowalska	kowalska_martusia@wp.pl	7
4	13	Mateusz Grubba	mati_gruby@gmail.com	7
5	12	Mikołaj Święty	swiety_mikolaj@laponia.pl	4

6.2. Stali_klienci

Widok Stali_klienci jest w założeniu bardzo podobny do poprzedniego. Tym razem jednak dotyczy tych, którzy zamówienia składają. Dzięki niemu wiemy, którzy klienci składają w sklepie najwięcej zamówień, a co za tym idzie w przyszłości możemy takiej osobie zaproponować atrakcyjne ceny produktów czy też specjalne oferty.

```
CREATE VIEW Stali_klienci AS
SELECT Klienci.Klient_ID, Klienci.Imię + ' ' + Klienci.Nazwisko AS KLIENT,
Klienci.Mail as Mail_klienta, COUNT(Zamówienia.Data_zamówienia) AS Ilość_zamówień
FROM Zamówienia INNER JOIN
      Klienci ON Zamówienia.ID_klienta = Klienci.Klient_ID
GROUP BY Klienci.Imię, Klienci.Nazwisko, Klienci.Klient_ID, Klienci.Mail

select * from Stali_klienci
order by Ilość_zamówień desc
```

	Klient_ID	KLIENT	Mail_klienta	Ilość_zamówień
1	3	Michał Piotrkowski	mich.pot@o2.pl	8
2	8	Eryk Martkowski	eryk_martkowski@wp.pl	6
3	1	Jakub Lemka	kuba.lemka@wp.pl	5
4	4	Karolina Kowalska	karolinka@buziaczek.pl	4
5	6	Michalina Wiejska	michalinka_cytrynka@gmail.com	4
6	7	Natasza Pietrova	russia_natasza@yahoo.com	4
7	2	Jakub Kamiński	jakub_kam@gmail.com	2
8	5	Natalia Wtorek	natalia_wtorek@wp.pl	1

6.3. Pracownik_na_minus

Widok pracownik na minus tym razem „wyróżnia” tych co na pochwałę nie zasługują a wręcz przeciwnie – zasługują na naganę. Wyświetla on wszystkich pracowników, którzy nie wykonali danego zamówienia, nie doprowadzili go do końca. Dodane zostało pole Dni_od_zamówienia, by od razu zwrócić uwagę na liczbę dni jakie minęły od złożenia danego zamówienia. Oczywiście jeśli ta liczba jest mała, np. mniej niż 7 dni, dany pracownik wciąż ma szansę wywiązać się z zamówienia w miarę w terminie. Tym razem zamieszczony został telefon pracownika, by kontakt z nim był jak najszybszy.

```
Create view Pracownik_na_minus as

SELECT      Pracownicy.Pracownik_ID,
Pracownicy.Imię + ' ' + Pracownicy.Nazwisko as Pracownik,
Zamówienia.Zamówienie_ID, Zamówienia.Data_zamówienia,
Pracownicy.Telefon as Telefon_pracownika,Zamówienia.Zrealizowano,
DATEDIFF(dd, zamówienia.Data_zamówienia,GETDATE()) as Dni_od_zamówienia
FROM
      Pracownicy INNER JOIN
      Zamówienia ON Pracownicy.Pracownik_ID = Zamówienia.ID_pracownika
WHERE      (Zamówienia.Zrealizowano = 'NIE')
order by Dni_od_zamówienia desc
```

100 % <

Messages

Command(s) completed successfully.

```
select * from Pracownik_na_minus
order by Dni_od_zamówienia desc
```

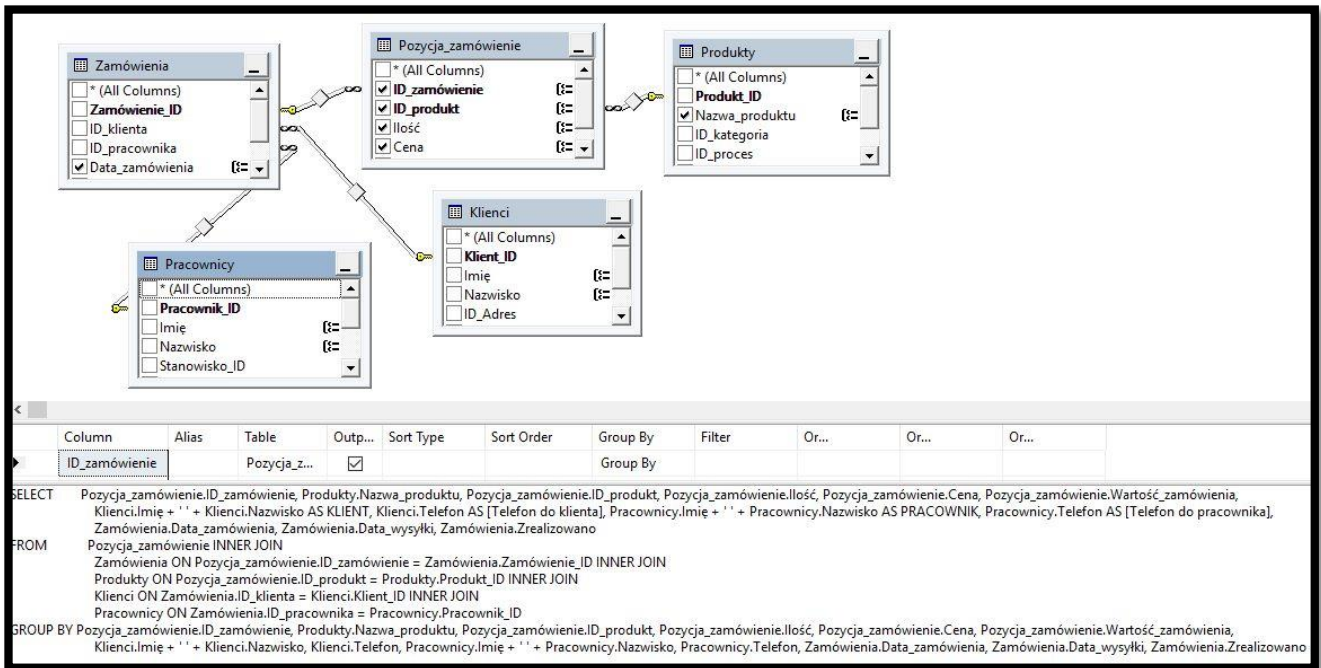
100 % <

Results Messages

	Pracownik_ID	Pracownik	Zamówienie_ID	Data_zamówienia	Telefon_pracownika	Zrealizowano	Dni_od_zamówienia
1	13	Mateusz Grubba	35	2015-01-22	523532342	NIE	365
2	3	Marta Kowalska	16	2015-08-19	212131131	NIE	156
3	3	Marta Kowalska	22	2015-09-28	212131131	NIE	116
4	2	Michalina Pelka	32	2016-01-18	777226172	NIE	4
5	2	Michalina Pelka	33	2016-01-19	777226172	NIE	3
6	13	Mateusz Grubba	34	2016-01-21	523532342	NIE	1

6.4. Szczegóły zamówień

Widok szczegóły zamówień zbiera informacje aż z pięciu tabel: Pozycja_zamówienie, Zamówienia, Produkty, Klienci, Pracownicy. Wyświetla nam wszystkie szczegółowe dane dotyczące każdego zamówienia. Umożliwia kontakt zarówno z klientem jak i pracownikiem w razie wystąpienia jakiś problemów z zamówieniem. Oczywiście poprzez ORDER BY można zawsze wyświetlić na samej górze te zamówienia, które nie są jeszcze zrealizowane.



select * from Szczegóły_zamówień												
ID_zamówienie	Nazwa_produktu	ID_produkt	Ilość	Cena	Wartość_zamówienia	KLIENT	Telefon do klienta	PRACOWNIK	Telefon do pracownika	Data_zamówienia	Data_wysyłki	Zrealizowano
1	Szampon miętowy	1	30	20,00	600,00	Michał Piotrkowski	888222333	Michalina Pelka	777226172	2015-01-10	2015-01-11	TAK
2	Odżywka do włosów farbowanych	9	30	45,00	1350,00	Michał Piotrkowski	888222333	Michalina Pelka	777226172	2015-01-10	2015-01-11	TAK
3	Zel w sprayu	26	10	88,00	880,00	Jakub Kamiński	777666555	Michalina Pelka	777226172	2015-02-02	2015-02-05	TAK
4	Szampon miętowy	1	35	20,00	700,00	Natasza Pietrowa	501983723	Katarzyna Miotke	421325223	2015-03-02	2015-03-11	TAK
5	Szampon owoce leśne	3	85	20,00	1700,00	Natasza Pietrowa	501983723	Katarzyna Miotke	421325223	2015-03-02	2015-03-11	TAK
6	Balsam do włosów długich	21	15	44,00	660,00	Natasza Pietrowa	501983723	Katarzyna Miotke	421325223	2015-03-02	2015-03-11	TAK
7	Balsam regenerujący	22	15	47,00	705,00	Natasza Pietrowa	501983723	Katarzyna Miotke	421325223	2015-03-02	2015-03-11	TAK
8	Pasta dla kobiet	29	10	30,00	300,00	Natasza Pietrowa	501983723	Katarzyna Miotke	421325223	2015-03-02	2015-03-11	TAK
9	Crema T & L	10	100	15,50	1550,00	Natasza Pietrowa	501983723	Katarzyna Miotke	421325223	2015-04-08	2015-05-01	TAK

7. Funkcje

7.1. Funkcje tabelowe

7.1.1. Klient_info

Funkcja tabelowa klient info wyświetla podstawowe informacje o wpisanym kliencie takie jak jego ID, imię i nazwisko, miejscowość, telefon, mail i ilość złożonych zamówień (pobrana z widoku Stali_klienci). Jest to w skrócie prosta i szybka wyszukiwarka każdego zapisanego klienta po jego imieniu i nazwisku. Umożliwia to zdobycie szybkiego kontaktu z nim czy też sprawdzenie ilości złożonych zamówień.

```
CREATE FUNCTION Klient_info (@KLIENT nvarchar(60))
RETURNS TABLE
AS
RETURN
(
SELECT      Klienci.Klient_ID, Klienci.Imię + ' ' + Klienci.Nazwisko AS KLIENT, Adresy.Miejscowość, Klienci.Telefon,
Klienci.Mail, Stali_klienci.Ilość_zamówień
FROM        Klienci INNER JOIN
            Adresy ON Klienci.ID_Adres = Adresy.Adres_ID INNER JOIN
            Stali_klienci ON Klienci.Klient_ID = Stali_klienci.Klient_ID
WHERE (KLIENT = @KLIENT)
)
```

Messages
Command(s) completed successfully.

```
use BD_Lemka_projekt
select * from Klient_info('Jakub Lemka')
```

100 %

Results Messages

	Klient_ID	KLIENT	Miejscowość	Telefon	Mail	Ilość_zamówień
1	1	Jakub Lemka	Gdynia	508176844	kuba.lemka@wp.pl	5

7.1.2. Pracownik_info

Funkcja pracownik_info jest niemalże 'kopią' funkcji klient_info. Tym razem pokazywane są podstawowe dane dotyczące wprowadzonego pracownika.

```
CREATE FUNCTION Pracownik_info (@PRACOWNIK nvarchar(60))
RETURNS TABLE
AS
RETURN
(
SELECT      Pracownicy.Pracownik_ID, Pracownicy.Imię + ' ' + Pracownicy.Nazwisko AS PRACOWNIK,
Stanowisko.Nazwa_stanowiska, Adresy.Miejscowość, Pracownicy.Telefon, Pracownicy.Mail
FROM        Pracownicy INNER JOIN
            Adresy ON Pracownicy.ID_Adres = Adresy.Adres_ID INNER JOIN
            Stanowisko ON Pracownicy.Stanowisko_ID = Stanowisko.Stanowisko_ID
WHERE (Pracownicy.Imię + ' ' + Pracownicy.Nazwisko = @PRACOWNIK)
)

SELECT * FROM Pracownik_info('Jakub Lemka')
```

100 %

Results Messages

	Pracownik_ID	PRACOWNIK	Nazwa_stanowiska	Miejscowość	Telefon	Mail
1	4	Jakub Lemka	Infomatyk	Gdynia	311253253	kuba.lemka@wp.pl

7.1.3. KategoriaProduktu

Funkcja KategoriaProduktu wyświetli nam wszystkie produkty z danej kategorii po jej wpisaniu. Dowiemy się też w jakiej fazie jest dany produkt, poznamy jego sugerowaną cenę.

```
CREATE FUNCTION KategoriaProduktu (@Kategoria nvarchar(50))
RETURNS TABLE
AS
RETURN
(
    SELECT      Produkty.Produkt_ID, Produkty.Nazwa_produktu, Kategoria.Nazwa_kategorii, Proces.Nazwa_procesu
    FROM        Produkty INNER JOIN
                Kategoria ON Produkty.ID_kategoria = Kategoria.Kategoria_ID INNER JOIN
                Proces ON Produkty.ID_proces = Proces.ID_Procesu
    WHERE      (Kategoria.Nazwa_kategorii = @Kategoria)
)|
```

00 % <

Messages

Command(s) completed successfully.

select * from KategoriaProduktu ('Zel')

100 % <

Results Messages

	Produkt_ID	Nazwa_produktu	Nazwa_kategorii	Nazwa_procesu	Cena_sugerowana
1	23	Zel super mocny Pudzian	Zel	Produkt GOTOWY	33,00
2	24	Zel o lekkim działaniu	Zel	Produkt GOTOWY	24,00
3	25	Zel dla Pań o krótkich włosach	Zel	Produkt GOTOWY	33,00
4	32	Zel z Coca Coli	Zel	Testy produktu	0,00

7.1.4. Szczegóły_danego_zamówienia

Funkcja Szczegóły_danego_zamówienia jest w prostych słowach rozwinięciem widoku Szczegóły_zamówień. W przypadku funkcji możliwe jest wprowadzenie ID dowolnego zamówienia a otrzymamy wszelkie potrzebne informacje na jego temat.

```
Create function Szczegóły_danego_zamówienia (@ID_Zamówienia int)
RETURNS TABLE
AS
RETURN
(
    SELECT      Pozycja_zamówienie.ID_zamówienie, Produkty.Nazwa_produktu, Pozycja_zamówienie.ID_produkt,
    Pozycja_zamówienie.Ilość, Pozycja_zamówienie.Cena, Pozycja_zamówienie.Wartość_zamówienia,
                Klienci.Imię + ' ' + Klienci.Nazwisko AS KLIENT, Klienci.Telefon AS [Telefon do klienta],
                Pracownicy.Imię + ' ' + Pracownicy.Nazwisko AS PRACOWNIK, Pracownicy.Telefon AS [Telefon do pracownika],
                Zamówienia.Data_zamówienia, Zamówienia.Data_wysyłki, Zamówienia.Zrealizowano
    FROM        Pozycja_zamówienie INNER JOIN
                Zamówienia ON Pozycja_zamówienie.ID_zamówienie = Zamówienia.Zamówienie_ID INNER JOIN
                Produkty ON Pozycja_zamówienie.ID_produkt = Produkty.Produkt_ID INNER JOIN
                Klienci ON Zamówienia.ID_klienta = Klienci.Klient_ID INNER JOIN
                Pracownicy ON Zamówienia.ID_pracownika = Pracownicy.Pracownik_ID
    WHERE (Pozycja_zamówienie.ID_zamówienie=@ID_Zamówienia)
)

SELECT * FROM Szczegóły_danego_zamówienia(11)
```

SELECT * FROM Szczegóły_danego_zamówienia(11)

	ID_zamówienie	Nazwa_produktu	ID_produkt	Ilość	Cena	Wartość_zamówienia	KLIENT	Telefon do klienta	PRACOWNIK	Telefon do pracownika	Data_zamówienia	Data_wysyłki	Zrealizowano
1	11	Szampon do włosów blond	6	50	29,90	1495,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
2	11	Szampon do włosów ciemnych	7	50	29,90	1495,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
3	11	Szampon do włosów rudych	8	1	100,00	100,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
4	11	Odżywka do włosów farbowanych	9	20	45,00	900,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
5	11	Odżywka na co dzień	10	200	19,90	3980,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
6	11	Lakier o mocy utrwalaenia 1/6	13	45	30,00	1350,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
7	11	Lakier o mocy utrwalaenia 2/6	14	45	30,00	1350,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
8	11	Lakier o mocy utrwalaenia 3/6	15	45	30,00	1350,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
9	11	Lakier o mocy utrwalaenia 4/6	16	45	30,00	1350,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
10	11	Lakier o mocy utrwalaenia 5/6	17	45	30,00	1350,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
11	11	Lakier o mocy utrwalaenia 6/6	18	45	30,00	1350,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
12	11	Balsam do włosów długich	21	34	44,00	1496,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
13	11	Balsam regenerujący	22	100	46,00	4600,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
14	11	Zel super mocny Pudzian	23	46	33,00	1518,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK
15	11	Zel o lekkim działaniu	24	73	20,00	1460,00	Michał Piotrkowski	888222333	Marta Kowalska	212131131	2015-07-22	2015-07-30	TAK

7.2. Funkcje skalarne

7.2.1. Staż_pracy

Funkcja staż_pracy dotyczy pracowników. Wyświetla ona liczbę dni, jakie minęły od czasu dołączenia danego pracownika do zespołu laboratorium. Może być ona bardzo przydatna, gdy ktoś z zarządu pragnie zastanowić się nad przyszłością danego pracownika. Często staż pracy jest w takim wypadku bardzo ważny. Danego pracownika szukamy po jego ID.

```
use BD_Lemka_projekt
go
CREATE FUNCTION staż_pracy (@id INT)
RETURNS INT
AS
BEGIN
    DECLARE @dzisiaj AS DATE
    DECLARE @roznica AS INT
    DECLARE @dolaczenie AS DATE
    SET @dzisiaj = GETDATE()
    SELECT @dolaczenie = Data_zatrudnienia from Pracownicy where @id = pracownicy.Pracownik_ID
    SET @roznica = DATEDIFF (DD, @dolaczenie, @dzisiaj)
    RETURN @Roznica
END
GO

select dbo.staż_pracy(2)
```

0 %

Results Messages

	(No column name)
1	1462

7.2.2. Ilość_zamówień

Funkcja skalarna Ilość_zamówień zwróci nam jak sama nazwa wskazuje ilość zamówień wykonanych przez danego klienta. Wprowadzamy jego ID.

```
CREATE FUNCTION Ilość_zamówień (@kID int)
returns int
as
begin
    declare @xx int
    select @xx = count (*) from zamówienia where ID_klienta = @kID
    RETURN @xx
END
```

100 %

Messages

Command(s) completed successfully.

```
select dbo.ilość_zamówień(1)
select dbo.ilość_zamówień(2)
select dbo.ilość_zamówień(5)
```

100 %

Results Messages

	(No column name)
1	5

	(No column name)
1	2

	(No column name)
1	1

8. Triggery

8.1. BLOKADA_NIEGOTOWYCH

Wymyślony trigger ma za zadanie uniemożliwić wprowadzenie do zamówienia produktów, które wciąż nie są gotowe, czyli są w fazie tworzenia bądź testowania. Jako, że produkt gotowy to taki, którego ID_Procesu wynosi zawsze 1, to tę właściwość wykorzystałem przy tworzenia wyzwalacza. Zacząłem jednak od stworzenia widoku. Etapy pokazane poniżej:

The screenshot shows two SQL queries being executed in SQL Server Enterprise Manager. The first query creates a view named BLOKADA_NIEGOTOWYCH_PRODUKTÓW. The second query creates a trigger named BLOKADA_NIEGOTOWYCH on the table Pozycja_zamówienie.

```
create view BLOKADA_NIEGOTOWYCH_PRODUKTÓW as
SELECT      Pozycja_zamówienie.ID_produkt, Produkty.ID_proces
FROM        Pozycja_zamówienie INNER JOIN
            Produkty ON Pozycja_zamówienie.ID_produkt = Produkty.Produkt_ID
```

	ID_produkt	ID_proces
1	1	1
2	9	1

```
USE [BD_Lemka_projekt]
GO

create trigger dbo.BLOKADA_NIEGOTOWYCH on
[dbo].[Pozycja_zamówienie]
after insert
as
declare @xx int, @ID_Produkt int
select @ID_Produkt=inserted.ID_Produkt from INSERTED
select @xx=ID_proces from dbo.BLOKADA_NIEGOTOWYCH_PRODUKTÓW
IF @xx=1
PRINT 'Produkt gotowy'
ELSE
BEGIN
PRINT 'Produkt niegotowy, zamówienie niemożliwe'
ROLLBACK
END
|
```

Command(s) completed successfully.

The screenshot shows two SQL insert statements being executed. The first insert is successful, and the second insert is aborted due to the trigger.

```
insert into Pozycja_zamówienie
values ('36','8','25','29')
```

Produkt gotowy
(1 row(s) affected)

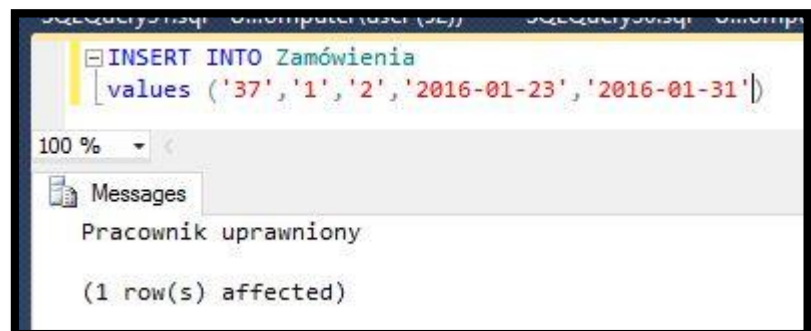
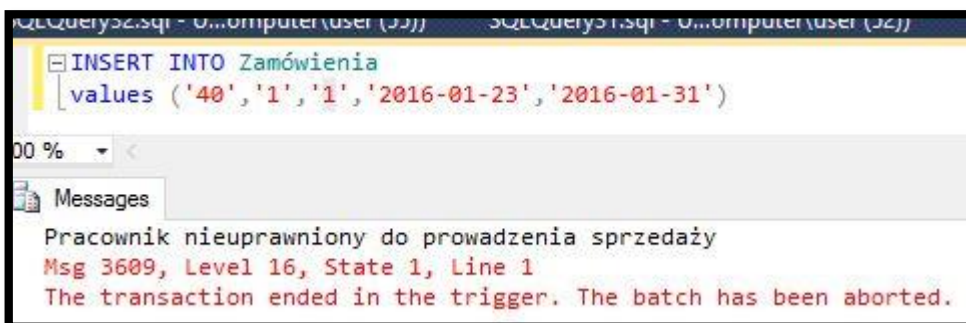
```
insert into Pozycja_zamówienie
values ('36','31','25','29')
```

Produkt niegotowy, zamówienie niemożliwe
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.

8.2. Uprawnienia

Jednym z założeń bazy danych jest to, iż nie wszyscy pracownicy są odpowiedzialni za 'opiekę' nad zamówieniami. By było to pod kontrola stworzyłem trigger, który daje takie uprawnienia tylko określonym pracownikom.

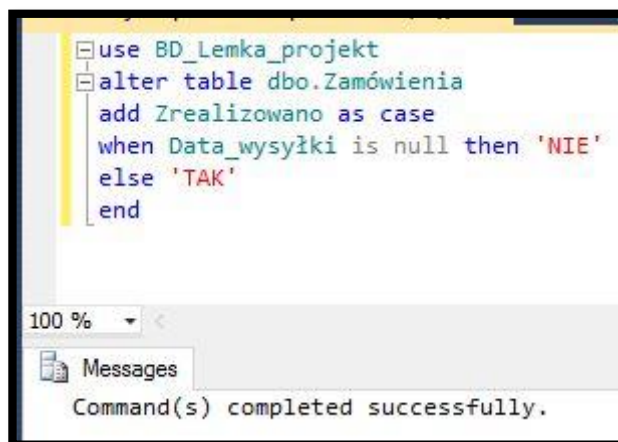
```
USE [BD_Lemka_projekt_zaliczeniowy]
GO
/***** Object: Trigger [dbo].[Uprawnienia]    Script Date: 2016-01-27 14:29:07 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[Uprawnienia] ON [dbo].[Zamówienia]
AFTER INSERT AS
DECLARE @ww int
SELECT @ww=ID_pracownika from dbo.Zamówienia
IF @ww in (2,3,5,9,12,13)
PRINT 'Pracownik uprawniony'
ELSE
BEGIN
PRINT 'Pracownik nieuprawniony do prowadzenia sprzedaży'
ROLLBACK
END
```



9. Inne elementy

Baza danych posiada parę niewidocznych na pierwszy rzut oka elementów. Niektóre z nich przewinęły się w poprzednich rozdziałach, niektóre można było zauważyć na 'print screenach'. W bazie znajduje się jeden podwójny klucz główny – w tabeli Pozycja_zamówienie. Składa się on z pól ID_zamówienie oraz ID_produkt, co pozwala na zawarcie kilku produktów w jednym zamówieniu. W tej samej tabeli znajduje się pole Wartość_zamówienia, którego nie możemy sami edytować. Jest to bowiem iloczyn ilości i ceny. Typ takiego pola nazwany jest w programie SQL jako 'Computed'.

Podobnie w tabeli Zamówienia, pola Zrealizowano nie wypełniamy my. Jest ono automatycznie wypełniane, jeśli pojawi się data wysyłki w polu o takiej samej nazwie (Data_wysyłki). Gdy zamówienie zostanie wysłane, w polu pojawia się słowo 'TAK'. Gdy nie ma wpisanej daty wysyłki, widnieje 'NIE'. Większość pól w tabelach musi być wypełniona – nie jest możliwe pozostawienie ich pustych podczas wypełniania. W stworzonej bazie danych starałem się, by pola w tabelach zawierały najważniejsze informacje, stąd też na tylko pojedyncze z nich mogą pozostać puste.

A screenshot of a SQL Server Enterprise Manager window. The top pane shows a T-SQL query:

```
use BD_Lemka_projekt
alter table dbo.Zamówienia
add Zrealizowano as case
when Data_wysyłki is null then 'NIE'
else 'TAK'
end
```

 The bottom pane shows a status message: "Command(s) completed successfully." The window title is "Messages".

10. Podsumowanie

Podsumowując moją pracę nad projektem i nad stworzeniem bazy danych – jestem zadowolony z efektu końcowego. Była to moja pierwsza w pełni samodzielnie wykonana baza danych i jak to mówią – „pierwsze koty za płoty”. Człowiek najlepiej uczy się wykonując pracę samodzielnie i szukając pomocy w różnych źródłach. Pomimo napotkanych trudności, zwłaszcza na początku przy projektowaniu całej bazy, ale też na przykład przy tworzeniu wyzwalaczy, udało się większość z nich pokonać.

Choć jest to oczywiście bardzo prosta, i niezłożona baza danych, stanowi dobrą podstawę do bardziej rozbudowanej bazy dla laboratorium czy też hurtowni kosmetyków.