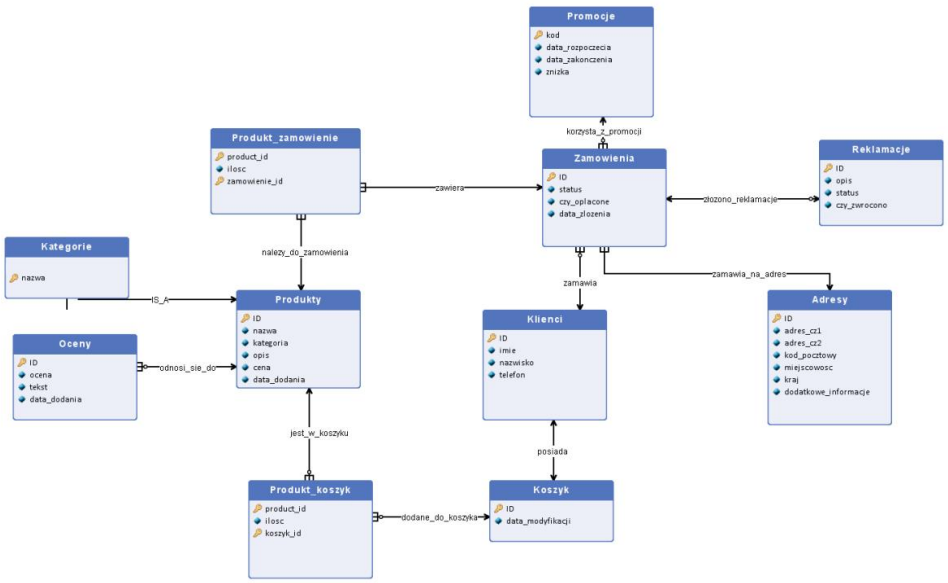
Bartosz Żylwis, 184477 18.12.2021

Sklep, towary w nim sprzedawane i klienci

1. Baza danych sklepu internetowego, wykorzystywana do wyświetlania produktów, ocen, składania zamówień i reklamacji.

2.



3. Opis zbioru encji (typy, klucze, ...)

Produkty			
Tabela przetrzymująca nazwy, kategorie, opisy i ceny wszystkich dostępnych produktów w sklepie. Odnosi się do typu produktów, nie do fizycznych egzemplarzy (np. produkt: Podręcznik historii Gdańska odnosi się do wszystkich fizycznych podręczników historii Gdańska dostępnych w sklepie) Nowe encje powstają, gdy sklep wprowadza nowe produkty do swojej oferty, lecz ze względy na archiwizację zamówień nigdy nie są usuwane. Liczebność: ok kilkadziesiąt tysięcy			
Nazwa	**Klucz główny**	**Typ/Dziedzina**	**Opis**
ID	**Tak**	Liczba naturalna	Unikalny identyfikator produktu
`ID INT IDENTITY(1,1) PRIMARY KEY,`			
nazwa	Nie	Tekst, max 50 znaków bez „!” i „?”	Nazwa produktu
`nazwa VARCHAR(50) NOT NULL CHECK (nazwa NOT LIKE '%!%' and nazwa NOT LIKE '%@%'),`			
kategoria	Nie	Tekst, pobrany z encji słownikowej	Nazwa kategorii pobrana z encji słownikowej Kategorie
`kategoria VARCHAR(50) REFERENCES Kategorie,`			

opis	Nie	Długi tekst, bez ograniczeń występowania znaków	Dokładny opis, wykorzystywany na stronie przeznaczonej danemu produktowi
opis text,			
cena	Nie	Dodatnia liczba zmiennoprzecinkowa w przedziale od 0.00 do 50 000.00	Cena produktu z dokładnością do dwóch miejsc po przecinku
cena DECIMAl(7,2) NOT NULL CHECK(cena >=0 AND cena <=50000),			
data_dodania	Nie	Data w formacie YYYY-MM-DD	Data dodanie produktu do sklepu
data_dodania DATE DEFAULT GETDATE()			

Klienci			
Tabela zawierająca wszystkich klientów sklepu. Nowe encje są tworzone podczas rejestracji klientów i nie są nigdy usuwane. Liczebność ok. kilkadziesiąt tysięcy			
Nazwa	**Klucz główny**	**Typ/Dziedzina**	**Opis**
ID	**Tak**	Liczba naturalna	Unikalny identyfikator klienta
ID INT IDENTITY(1,1) PRIMARY KEY,			
imie	Nie	Słowo, max. 40 liter, bez spacji	Imię klienta
imie varchar(40) CHECK (imie NOT LIKE '% %'),			
nazwisko	Nie	Słowo, max. 40 liter, bez spacji	Nazwisko klienta
nazwisko varchar(40) CHECK (nazwisko NOT LIKE '% %'),			
telefon	Nie	Tekst składający się z 9 cyfr	Dziewięcioznakowy numer telefonu zapisany jako tekst by nie pominąć np. początkowych zer
telefon varchar(9) CHECK(telefon LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')			

Produkt_zamowienie			
Zbiór encji przyporządkowujący produkty odpowiednim zamówieniom. Nowe encje powstają podczas dodawania produktów do zmówienia. Liczebność: ok kilkaset tysięcy			
Nazwa	**Klucz główny**	**Typ/Dziedzina**	**Opis**
product_id	**Tak**	Liczba naturalna	Identyfikator produktu
produkt_id int NOT NULL,			
ilosc	Nie	Liczba naturalna mniejsza, niż 1000	Liczba sztuk danego produktu
ilosc int not null CHECK(ilosc >= 0 AND ilosc <= 1000),			
zamowienie_id	**Tak**	Liczba naturalna	Identyfikator zamówienia
zamowienie_id int NOT NULL,			

Zamowienia

Tabela zawierająca wszystkie zamówienia. Nowe encje są tworzone po zatwierdzeniu produktów z koszyka lub bezpośrednio ze strony produktu po zatwierdzeniu chęci kupna. Liczebność: ok kilkaset tysięcy

Nazwa	Klucz główny	Typ/Dziedzina	Opis
ID	**Tak**	Liczba naturalna	Unikalny identyfikator zamówienia
`ID int IDENTITY(1,1) PRIMARY KEY,`			
status	Nie	Tekst, jedno z: „przyjete” , ”zapakowane” , ”wyslane” , ”odebrane”	Obecny status zamówienia
`status varchar(15) DEFAULT('przyjete') CHECK(status in ('przyjete', 'zapakowane', 'wyslane', 'odebrane')),`			
czy_oplacone	Nie	Bool, prawda / fałsz	Informacja czy klient opłaci to zamówienie
`czy_oplacone bit default(0),`			
data_zlozenia	Nie	Data z godziną w formacie YYYY-MM-DD hh:mm:ss	Dokładna data złożenia zamówienia
`data_zlozenia datetime DEFAULT GETDATE()`			

Adresy

Tabela zawierająca adresy. Nowe encje powstają podczas składania zamówień i nigdy nie są usuwane Liczebność: ok kilkadziesiąt tysięcy

Nazwa	Klucz główny	Typ/Dziedzina	Opis
ID	**Tak**	Liczba naturalna	Unikalny identyfikator adresu
`ID int IDENTITY(1,1) PRIMARY KEY,`			
adres_cz1	Nie	Tekst, składający się z max 40 liter i cyfr bez znaków specjalnych	Pierwsza część adresu (ulica i numer budynku)
`adres_cz1 varchar(40) not null CHECK(adres_cz1 NOT LIKE '%[^A-Z/.ĄŻŹĘĆŃÓŁ 0-9]%'),`			
adres_cz2	Nie	Tekst, składający się z max 40 liter i cyfr bez znaków specjalnych	Druga część adresu (nr lokalu)
`adres_cz2 varchar(40) CHECK(adres_cz2 NOT LIKE '%[^A-Z/.ĄŻŹĘĆŃÓŁ 0-9]%'),`			
kod_pocztowy	Nie	Tekst składający się z 5 cyfr w formacie 00-000 np. „16-500”	Pięciocyfrowy tekst będący kodem pocztowym
`kod_pocztowy char(6) not null check (kod_pocztowy LIKE '[0-9][0-9]-[0-9][0-9][0-9]'),`			

miejscowosc	Nie	Tekst składający się z max. 30 liter, dozwolone są spacje	Nazwa miejscowości
`miejscowosc varchar(30) CHECK(miejscowosc NOT LIKE '%[^A-Z/.ĄŻŹĘĆŃÓŁ 0-9]%'),`			
kraj	Nie	Tekst składający się z max. 58 liter, dozwolone są spacje	Nazwa kraju
`kraj varchar(58) CHECK(kraj NOT LIKE '%[^A-ZĄŻŹĘĆŃÓŁ 0-9]%'),`			
dodatkowe_informacje	Nie	Tekst składający się z max. 100 znaków	Opcjonalne dodatkowe informacje odnośnie adresu
`dodatkowe_informacje varchar(100)`			

Kategorie

Encja słownikowa przetrzymująca kategorie. Dane są dodawane podczas tworzenia systemu i nigdy nie są usuwane. Liczebność: ok kilkadziesiąt

Nazwa	Klucz główny	Typ/Dziedzina	Opis
nazwa	**Tak**	Tekst składający się z max. 50 liter i spacji	Nazwa kategorii
`nazwa varchar(50) PRIMARY KEY CHECK(nazwa NOT LIKE '%[^A-Z ]%')`			

Promocje

Tabela zawierająca wszystkie kody promocyjne wraz z terminami ważności. Nowe encje są dodawane średnio raz w miesiącu i usuwane rok po utracie ważności. Liczebność: ok kilkadziesiąt

Nazwa	Klucz główny	Typ/Dziedzina	Opis
kod	**Tak**	Słowo składające się z max. 10 znaków, wyłącznie duże litery i cyfry	Unikalny kod składający się z liter i cyfr identyfikujący promocję
`kod varchar(10) PRIMARY KEY CHECK(kod NOT LIKE '%[^A-Z0-9]%'),`			
data_rozpoczecia	Nie	Data z godziną w formacie YYYY-MM-DD hh:mm:ss	Data rozpoczęcia ważności kodu promocyjnego
`data_rozpoczecia DATETIME DEFAULT GETDATE(),`			
data_zakonczenia	Nie	Data z godziną w formacie YYYY-MM-DD hh:mm:ss	Termin ważności kodu promocyjnego
`data_zakonczenia DATETIME DEFAULT (GETDATE()+30),`			

znizka	Nie	Liczba zmiennoprzecinkowa w przedziale od 0 do 1 z dokładnością do dwóch miejsc po przecinku	Wartość przedstawiająca wielkość zniżki w przedziale od 0 do 1 np. 0.5 oznacza zniżkę 50%

```
znizka DECIMAl(4,2) NOT NULL CHECK(znizka >=0 AND znizka <=1)
```

Reklamacje

Tabela zawierająca reklamacje przyporządkowane odpowiedniemu zamówieniu. Nowe encje powstają, jeżeli klient sklepu zgłosi reklamację, nigdy nie są usuwane. Liczebność: ok 10 000

Nazwa	Klucz główny	Typ/Dziedzina	Opis
ID	**Tak**	Liczba naturalna	Unikalny identyfikator reklamacji.

```
ID int IDENTITY(1,1) PRIMARY KEY,
```

opis	Nie	Długi tekst, bez ograniczeń występowania znaków	Opis przyczyny reklamacji złożonej przez klienta.

```
opis text not null,
```

status	Nie	Tekst, jedno z: „rozpatrywana” , „odrzucona” , „przyjeta”	Status reklamacji.

```
status varchar(15) default('rozpatrywana') CHECK(status in ('rozpatrywana', 'odrzucona', 'przyjeta')),
```

czy_zwrocono	Nie	Bool, prawda / fałsz	Informacja czy dokonano zwrotu pieniędzy.

```
czy_zwrocono bit default(0)
```

Koszyk

Tabela zawierająca koszyki poszczególnych klientów. Nowa koszyk jest tworzony podczas dodawania klienta i nigdy nie zastaje usunięty. Tabela ta występuje, by klient logując się z różnych urządzeń mógł nadal korzystać z tego samego koszyka. Liczebność: równa liczbie klientów – ok kilkadziesiąt tysięcy

Nazwa	Klucz główny	Typ/Dziedzina	Opis
ID	**Tak**	Liczba Naturalna	Unikalny identyfikator koszyka

```
ID int  IDENTITY(1,1) PRIMARY KEY,
```

data_modyfikacji	Nie	Data z godziną w formacie YYYY-MM-DD hh:mm:ss	Data ostatniej modyfikacji(dodanie lub usunięcie przedmiotu z koszyka)

```
data_modyfikacji DATETIME
```

Produkt_koszyk			
Zbiór encji asocjacyjnych przyporządkowujący produkty odpowiednim koszykom. Powstaje podczas dodawania produktów do koszyka i jest usuwane w momencie zmiany wartości ilości na 0. Liczebność: ok kilkaset tysięcy			
Nazwa	**Klucz główny**	**Typ/Dziedzina**	**Opis**
product_id	**Tak**	Liczba naturalna	Identyfikator produktu
`produkt_id int not null,`			
ilosc	Nie	Liczba naturalna mniejsza niż 1000	Liczba sztuk danego produktu
`ilosc int not null CHECK(ilosc >=0 AND ilosc < 1000),`			
koszyk_id	**Tak**	Liczba naturalna	Identyfikator koszyka
`koszyk_id int not null,`			

Oceny			
Anonimowe opinie klientów na temat poszczególnych produktów. Nowe encje powstają, gdy użytkownik postanowi wystawić ocenę kupionemu produktowi i nie jest nigdy usuwana. Liczebność: ok kilkadziesiąt tysięcy			
Nazwa	**Klucz główny**	**Typ/Dziedzina**	**Opis**
ID	**Tak**	Liczba naturalna	Unikalny identyfikator oceny.
`ID int IDENTITY(1,1) PRIMARY KEY,`			
ocena	Nie	Liczba naturalna w zakresie 1-5	Wystawiona ocena w przedziale od 1 do 5, gdzie 5 oznacza najlepszą, natomiast 1 najgorszą.
`ocena int not null CHECK (ocena >= 1 AND ocena <=5),`			
tekst	Nie	Tekst składający się z max. 100 znaków	Opcjonalny tekst opisujący przedmiot lub argumentujący przyznaną ocenę.
`tekst varchar (100),`			
data_dodania	Nie	Data z godziną w formacie YYYY-MM-DD hh:mm:ss	Data dodania oceny.
`data_dodania DATETIME DEFAULT GETDATE()`			

4. Schemat relacyjnej bazy danych

Produkty (**ID**, nazwa, kategoria, opis, cena, data_dodania)

(ID) Key

```
ID  INT IDENTITY(1,1) PRIMARY KEY,
```

(kategoria) REF Kategorie

```
ALTER TABLE Produkty ADD CONSTRAINT nalezy_do_kategorii FOREIGN KEY (kategoria) REFERENCES Kategorie(nazwa)
```

Klienci (**ID**, imie, nazwisko, telefon, koszyk_id)

(ID) Key

```
ID INT IDENTITY(1,1) PRIMARY KEY,
```

(koszyk_id) REF Koszyk

```
ALTER TABLE Klienci ADD CONSTRAINT posiada FOREIGN KEY (koszyk_id) REFERENCES Koszyk(ID)
```

Koszyk (**ID**, data_modyfikacji)

(ID) Key

```
ID int  IDENTITY(1,1) PRIMARY KEY,
```

Produkt_koszyk (**produkt_id**, **koszyk_id**, ilosc)

(produkt_id, koszyk_id) Key

```
PRIMARY KEY (produkt_id, koszyk_id)
```

(produkt_id) REF Produkty

```
ALTER TABLE Produkt_koszyk ADD CONSTRAINT jest_w_koszyku FOREIGN KEY (produkt_id) REFERENCES Produkty(ID)
```

(koszyk_id) REF Koszyk

```
ALTER TABLE Produkt_koszyk ADD CONSTRAINT dodane_do_koszyka FOREIGN KEY (koszyk_id) REFERENCES Koszyk(ID)
```

Oceny (**ID**, ocena, tekst, data_dodania, produkt_id)

(ID) Key

```
ID int IDENTITY(1,1) PRIMARY KEY,
```

(produkt_id) REF Produkty

```
ALTER TABLE Oceny ADD CONSTRAINT odnosi_sie_do FOREIGN KEY (produkt_id) REFERENCES Produkty(ID)
```

Kategorie (**ID**, nazwa)

(ID) Key

```
nazwa varchar(50) PRIMARY KEY CHECK(nazwa NOT LIKE '%[^A-Z ]%')
```

Zamowienia (**ID**, status, czy_oplacone, data_zlozenia, adres_id, klient_id, kod_promocji)

(ID) Key

```
ID int IDENTITY(1,1) PRIMARY KEY,
```

(adres_id) REF Adresy

```
ALTER TABLE Zamowienia ADD CONSTRAINT zamawia_na_adres FOREIGN KEY (adres_id) REFERENCES Adresy(ID)
```

(klient_id) REF Klienci

```
ALTER TABLE Zamowienia ADD CONSTRAINT zamawia FOREIGN KEY (klient_id) REFERENCES Klienci(ID)
```

(kod_promocji) REF Promocje

```
ALTER TABLE Zamowienia ADD CONSTRAINT korzysta_z_promocji FOREIGN KEY (kod_promocji) REFERENCES Promocje(kod)
```

Produkty_zamowienie (**produkt_id**, **zamowienie_id**, ilosc)

(produkt_id, zamówienie_id) Key

```
PRIMARY KEY(produkt_id, zamowienie_id)
```

(produkt_id) REF Produkty

```
ALTER TABLE Produkt_zamowienie ADD CONSTRAINT nalezy_do_zamowienia FOREIGN KEY (produkt_id) REFERENCES Produkty(ID)
```

(zamówienie_id) REF Zamowienia

```
ALTER TABLE Produkt_zamowienie ADD CONSTRAINT zawiera FOREIGN KEY (zamowienie_id) REFERENCES Zamowienia(ID)
```

Promocje (**kod**, data_rozpoczecia, data_zakonczenia, znizka)
(kod) Key

```
kod varchar(10) PRIMARY KEY CHECK(kod NOT LIKE '%[^A-Z0-9]%'),
```

Reklamacje (**ID**, opis, status, czy_zwrocono. zamowienie_id)
(ID) Key

```
ID int IDENTITY(1,1) PRIMARY KEY,
```

(zamowienie_id) REF Zamowienia

```
ALTER TABLE Reklamacje ADD CONSTRAINT zlozono_reklamacje FOREIGN KEY (zamowienie_id) REFERENCES Zamowienia(ID)
```

Adresy (**ID**, adres_cz1, adres_cz2, kod_pocztowy, miejscowość, kraj, dodatkowe_informacje)
(ID) Key

```
ID int IDENTITY(1,1) PRIMARY KEY,
```

5. Szczegółowy opis utworzonych tabel pod kątem zastosowanych ograniczeń np. NOT NULL, UNIQUE, CHECK, DEFAULT, klucze …

```
CREATE TABLE Kategorie (
    nazwa varchar(50) PRIMARY KEY CHECK(nazwa NOT LIKE '%[^A-Z ]%')
)
```

nazwa posiada sprawdzenie (CHECK) by nazwy kategorii składały się wyłącznie z podstawowych liter alfabetu i spacji.

```
CREATE  TABLE Produkty (
    ID  INT IDENTITY(1,1) PRIMARY KEY,
    nazwa VARCHAR(50) NOT NULL CHECK (nazwa NOT LIKE '%!%' and nazwa NOT LIKE '%@%'),
    kategoria VARCHAR(50) REFERENCES Kategorie,
    opis text,
    cena DECIMAl(7,2) NOT NULL CHECK(cena >=0 AND cena <=50000),
    data_dodania DATE DEFAULT GETDATE()
)
```

ID posiada IDENTITY(1,1) w celu automatycznego przypisywania unikalnych kluczy głównych w tej tabeli zaczynając od numeru 1 i inkrementując kolejne o jeden.
nazwa nie może być pusta, gdyż uniemożliwiłoby to wyświetlanie listy produktów zrozumiałej przez użytkowników. Dodatkowo występuje tu sprawdzenie CHECK, które zapobiega wystąpieniom ‘!’ i ‘?’ zgodnie z założeniami projektu.
cena nie może być pusta, gdyż jest to jedna z najważniejszych atrybutów produktu i musi zawierać się w przedziale <0; 50 000> zgodnie z założeniami projektu.
data_dodania domyślnie jest ustawiana na obecną datę, gdyż pozwoli to na skrócenia zapytań dodających nowe rekordy.

```
CREATE TABLE Klienci (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    imie varchar(40) CHECK (imie NOT LIKE '% %'),
    nazwisko varchar(40) CHECK (nazwisko NOT LIKE '% %'),
    telefon varchar(9) CHECK(telefon LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
    koszyk_id int
)
```

ID posiada IDENTITY(1,1) w celu automatycznego przypisywania unikalnych kluczy głównych w tej tabeli zaczynając od numeru 1 i inkrementując kolejne o jeden.
imie posiada sprawdzenie, czy we wprowadzanym ciągu znaków nie występuje spacja, ponieważ pole to służy wyłącznie do przechowywania jednego imienia.
nazwisko posiada sprawdzenie, czy we wprowadzanym ciągu znaków nie występuje spacja, ponieważ pole to służy wyłącznie do przechowywania jednego nazwiska.
telefon posiada sprawdzenie czy wprowadzony ciąg znaków jest tekstem składającym się z 9 cyfr. Zapewnia nam to, że w tym polu będą przechowywane poprawne, polskie numery telefonów.

```
CREATE TABLE Produkt_zamowienie (
    produkt_id int NOT NULL,
    zamowienie_id int NOT NULL,
    ilosc int not null CHECK(ilosc >= 0 AND ilosc <= 1000),
    PRIMARY KEY(produkt_id, zamowienie_id)
)
```

produkt_id nie może być pusty, ponieważ jest wykorzystywany do zdefiniowania złożonego klucza głównego tabeli.
zamowienie_id nie może być puste, ponieważ jest wykorzystywane do zdefiniowania złożonego klucza głównego tabeli.
ilość nie może być pusta, ponieważ jeżeli jakiś produkt należy do zamówienia to jego ilość jest >= 1. Dodatkowy występuje sprawdzenie czy wprowadzona ilość jest liczbą z przedziały <0;1000> zgodnie z założeniami projektu. Liczy ujemne są niedopuszczalne, gdyż są niezgodne z przyjętą definicją ilości.

```
CREATE TABLE Zamowienia (
    ID int IDENTITY(1,1) PRIMARY KEY,
    status varchar(15) DEFAULT('przyjete') CHECK(status in ('przyjete', 'zapakowane', 'wyslane', 'odebrane')),
    czy_oplacone bit default(0),
    data_zlozenia datetime DEFAULT GETDATE(),
    adres_id int,
    klient_id int,
    kod_promocji varchar(10)
)
```

ID posiada IDENTITY(1,1) w celu automatycznego przypisywania unikalnych kluczy głównych w tej tabeli zaczynając od numeru 1 i inkrementując kolejne o jeden.
status domyślnie zostaje ustawiony na ‘przyjete’ i pole posiada sprawdzenie, czy zawiera się on w zdefiniowanej liście możliwych statusów zamówienia.
czy_oplacone pole domyślnie przyjmuje wartość 0 (false), ponieważ w większości przypadku po złożeniu zamówienia płatność nie będzie jeszcze zweryfikowana.
data_złożenia domyślnie ustawiania jest obecna data, gdyż zamówienie zostaje złożone w momencie zapisania w bazie danych

```
CREATE TABLE Adresy (
    ID int IDENTITY(1,1) PRIMARY KEY,
    adres_cz1 varchar(40) not null CHECK(adres_cz1 NOT LIKE '%[^A-Z/.ĄŻŹĘĆŃÓŁ 0-9]%'),
    adres_cz2 varchar(40) CHECK(adres_cz2 NOT LIKE '%[^A-Z/.ĄŻŹĘĆŃÓŁ 0-9]%'),
    kod_pocztowy char(6) not null CHECK (kod_pocztowy LIKE '[0-9][0-9]-[0-9][0-9][0-9]'),
    miejscowosc varchar(30) CHECK(miejscowosc NOT LIKE '%[^A-Z/.ĄŻŹĘĆŃÓŁ 0-9]%'),
    kraj varchar(58) CHECK(kraj NOT LIKE '%[^A-ZĄŻŹĘĆŃÓŁ 0-9]%'),
    dodatkowe_informacje varchar(100)
)
```

ID posiada IDENTITY(1,1) w celu automatycznego przypisywania unikalnych kluczy głównych w tej tabeli zaczynając od numeru 1 i inkrementując kolejne o jeden.
adres_cz1 nie może być pusty, ponieważ jest to jedno z najważniejszych pół w tabeli. Dodatkowo sprawdzane jest, czy adres nie posiada znaków specjalnych
adres_cz2 posiada sprawdzenie, czy nie występują znaki specjalne

kod_pocztowy nie może być pusty, ponieważ jest to jedna z najważniejszych pól tabeli. Dodatkowy występuje sprawdzenie poprawności kody, czyli wystąpienia dwóch liczb, znaku '-' i zakończenia trzema cyframi
miejscowosc posiada sprawdzenie, czy nie występują znaki specjalne
kraj posiada sprawdzenie, czy nie występują znaki specjalne

```
CREATE TABLE Promocje (
    kod varchar(10) PRIMARY KEY CHECK(kod NOT LIKE '%[^A-Z0-9]%'),
    data_rozpoczecia DATETIME DEFAULT GETDATE(),
    data_zakonczenia DATETIME DEFAULT (GETDATE()+30),
    znizka DECIMAl(4,2) NOT NULL CHECK(znizka >=0 AND znizka <=1)
)
```

kod posiada sprawdzeniec czy nie występują znaki spoza podstawowego alfabetu i cyfr.
data_rozpoczenia domyślnie przyjmuje wartość daty wstawiania rekordu
data_zakonczenia domyślnie przyjmuje wartość daty wstawiania rekordu + 30 dni
znizka nie może być pusta, ponieważ jest niezbędna do wyliczenia zniżki naliczanej przy użycie danego kodu. Dodatkowo występuje sprawdzenie, czy wartość jest z przedziału <0; 1> zgodnie z założeniami projektu (np. 0.5 oznacza 50% zniżki)

```
CREATE TABLE Reklamacje (
    ID int IDENTITY(1,1) PRIMARY KEY,
    opis text not null,
    status varchar(15) default('rozpatrywana') CHECK(status in ('rozpatrywana', 'odrzucona', 'przyjeta')),
    czy_zwrocono bit default(0),
    zamowienie_id int
)
```

ID posiada IDENTITY(1,1) w celu automatycznego przypisywania unikalnych kluczy głównych w tej tabeli zaczynając od numeru 1 i inkrementując kolejne o jeden.
opis nie może być pusty zgodnie z założeniami projektu
status domyślnie przyjmuje wartość 'pozpatrywana', gdyż każda nowa reklamacja musi zostać przeanalizowana przez pracownika. Dodatkowo występuje sprawdzenie, czy wartość zawiera się on w zdefiniowanej liście możliwych statusów reklamacji.
czy_zwrócono domyślnie przyjmuje wartość 0, gdyż po złożeniu reklamacji zwrot nie jest przyznawany automatycznie

```
CREATE TABLE Koszyk (
    ID int  IDENTITY(1,1) PRIMARY KEY,
    data_modyfikacji DATETIME
)
```

ID posiada IDENTITY(1,1) w celu automatycznego przypisywania unikalnych kluczy głównych w tej tabeli zaczynając od numeru 1 i inkrementując kolejne o jeden.

```
CREATE TABLE Produkt_koszyk (
    produkt_id int not null,
    koszyk_id int not null,
    ilosc int not null CHECK(ilosc >=0 AND ilosc < 1000),
    PRIMARY KEY (produkt_id, koszyk_id)
)
```

produkt_id nie może być pusty, ponieważ jest wykorzystywany do zdefiniowania złożonego klucza głównego tabeli.
zamowienie_id nie może być puste, ponieważ jest wykorzystywane do zdefiniowania złożonego klucza głównego tabeli.

ilość nie może być pusta, ponieważ jeżeli jakiś produkt należy do koszyka to jego ilość jest >= 1. Dodatkowy występuje sprawdzenie czy wprowadzona ilość jest liczbą z przedziały <0;1000> zgodnie z założeniami projektu. Liczy ujemne są niedopuszczalne, gdyż są niezgodne z przyjętą definicją ilości.

```
CREATE TABLE Oceny (
    ID int IDENTITY(1,1) PRIMARY KEY,
    ocena int not null CHECK (ocena >= 1 AND ocena <=5),
    tekst varchar (100),
    data_dodania DATETIME DEFAULT GETDATE(),
    produkt_id int
)
```

ID posiada IDENTITY(1,1) w celu automatycznego przypisywania unikalnych kluczy głównych w tej tabeli zaczynając od numeru 1 i inkrementując kolejne o jeden.
ocena nie może być pusta ponieważ jest to najważniejsze pole w tej tabeli i musi przyjmować wartości całkowite z przedziału <1; 5> zgodnie z założeniami projektu
data_dodania domyślnie jest ustawiana obecna data – data dodania nowego rekordu.