Obraz zawierający kreskówka, clipart, żółty

Opis wygenerowany automatycznie

BaZA DANYCH - PIEKARNIA

Paweł Wozignój

Spis treści

[Opis tła projektu 2](#_Toc168526838)

[Wymagania funkcjonalne i niefunkcjonalne 3](#_Toc168526839)

[Model pojęciowy (chen) 4](#_Toc168526840)

[Wykaz encji logicznych 6](#_Toc168526841)

[Wykaz atrybutów 8](#_Toc168526842)

[Wykaz związków 17](#_Toc168526843)

[Model logiczny 19](#_Toc168526844)

[Model fizyczny 19](#_Toc168526845)

[Wykaz tabel fizycznych 20](#_Toc168526847)

[Skrypty SQL tworzące strukturę tabel i powiązania(MS SERVER) 21](#_Toc168526848)

[Wykaz indeksów 24](#_Toc168526849)

[Skrypty tworzące indeksy 24](#_Toc168526850)

[Wykaz widoków 25](#_Toc168526851)

[Skrypty tworzące widoki 25](#_Toc168526852)

[Skrypty SQL rejestrujące przykładowe dane 29](#_Toc168526853)

[Wykaz procedur i funkcji do generowania raportów 33](#_Toc168526854)

[Skrypty tworzące procedury i funkcje do generowania raportów 33](#_Toc168526855)

[Wykaz funkcji i procedur do przeglądania danych 39](#_Toc168526856)

[Skrypty tworzące funkcje i procedury do przeglądania danych 39](#_Toc168526857)

[Wykaz procedur do masowej aktualizacji danych 45](#_Toc168526858)

[Skrypty tworzące procedury do masowej aktualizacji danych 45](#_Toc168526859)

[Skrypty tworzące procedury i triggery do modyfikacji danych 49](#_Toc168526860)

[Schemat bazy danych w MS SQL SERVER 74](#_Toc168526861)

[Skrypty SQL tworzące strukturę tabel, powiązania i rejestrujące przykładowe dane(ORACLE) 75](#_Toc168526862)

[Wnioski 88](#_Toc168526863)

[Podsumowanie 88](#_Toc168526864)

# Opis tła projektu

Projekt został stworzony dla rzeczywistej firmy piekarniczej, która potrzebowała efektywnego narzędzia do zarządzania swoimi operacjami. Baza danych piekarni stanowi kluczowy element w codziennej pracy, gromadząc istotne informacje związane z działalnością firmy oraz relacjami z klientami, dostawcami i produktami.

Firma piekarnicza specjalizuje się w produkcji różnorodnych wyrobów piekarskich i cukierniczych, a baza danych obejmuje szeroki zakres danych, w tym informacje o pracownikach, ich stanowiskach pracy, szkoleniach, wynagrodzeniach, dostawcach, dostawach składników, magazynie, składnikach, produktach, kategoriach produktów, opakowaniach, zamówieniach oraz pojazdach dostawczych.

Dzięki temu systemowi firma piekarnicza może skutecznie zarządzać swoimi zasobami, monitorować stany magazynowe, śledzić zamówienia oraz generować różnorodne raporty wspomagające podejmowanie decyzji biznesowych. Projekt ten nie tylko ułatwi codzienną pracę pracownikom piekarni, ale także może przyczynić się do poprawy efektywności operacyjnej i zwiększenia satysfakcji klientów. Ponadto, możliwe jest, że w przyszłości projekt zostanie rozszerzony lub wykorzystany przez inne firmy w podobnej branży, co może przynieść korzyści nie tylko dla tej konkretnej firmy, ale również dla szerszej społeczności biznesowej.

# Wymagania funkcjonalne i niefunkcjonalne

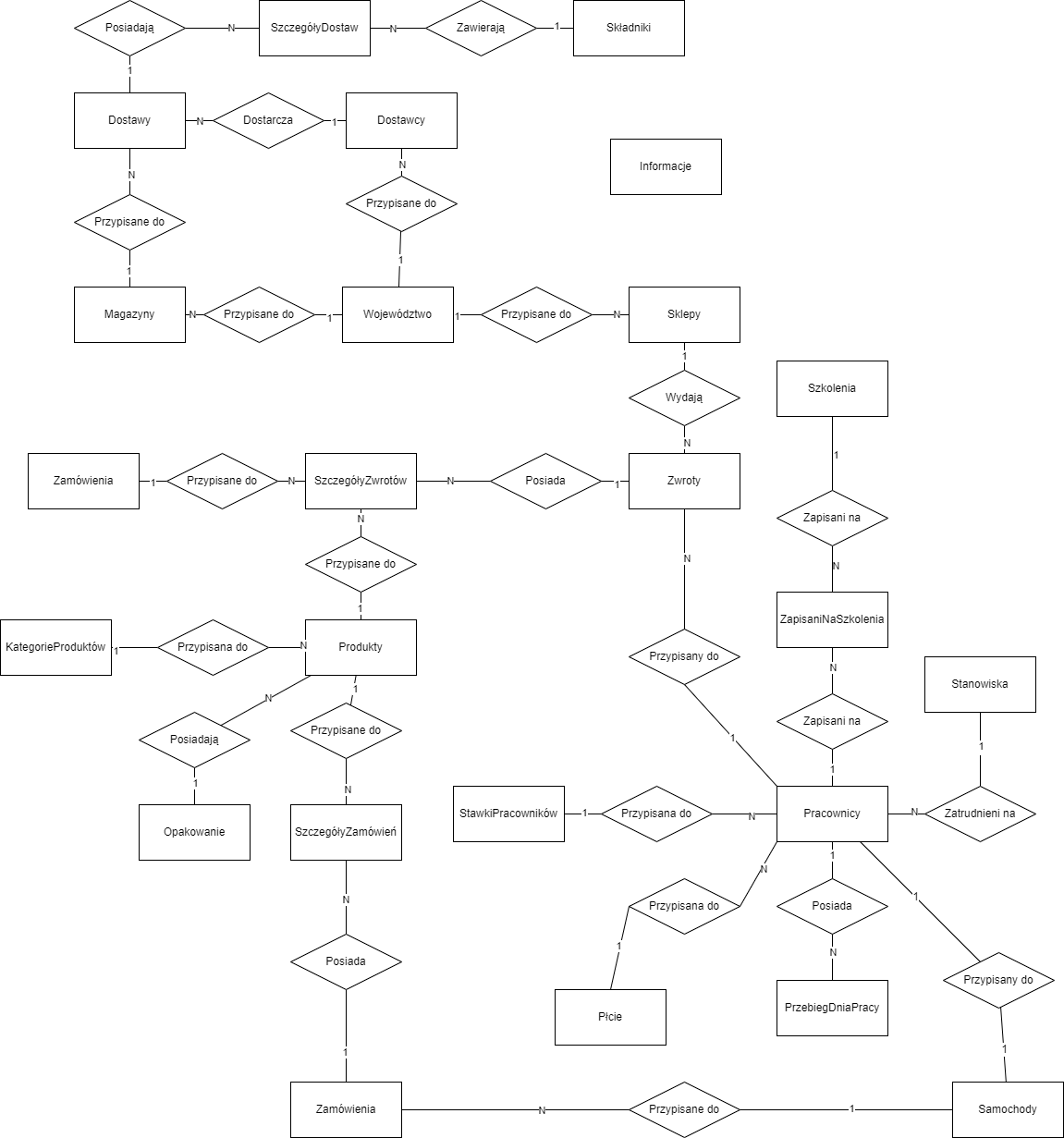
Funkcjonalne:

* Rejestracja sklepów:
  + Dodawanie nowych sklepów do bazy
  + Aktualizacja informacji o istniejących sklepach
  + Przypisywanie unikalnych identyfikatorów dla każdego sklepu
  + Usuwanie błędnie wprowadzonych sklepów
* Informacje o zwrotach
  + Zarejestrowanie informacji o nowych zwrotach oraz o ich szczegółach ( jakie produkty są zwracane)
  + Usuwanie niepotrzebnych zwrotów
  + Aktualizowanie danych na temat wybranego zwrotu
* Zarządzanie Pracownikami
  + Dodawanie nowych pracowników, ich danych osobowych oraz wynagrodzenia
  + Usuwanie pracowników
  + Aktualizacja danych pracownika jeżeli będzie potrzeba np. zwiększenia pensji pracownika;
* Rejestracja Dostaw
  + Dodawanie, Usuwanie, Aktualizowanie informacji na temat dostaw składników dostarczanych do piekarni
  + Zarejestrowanie informacji na temat dostawców, którzy realizują dostawy
  + Informacje na temat składników, które są dostarczane
  + Rejestracja informacji na temat magazynów
* Rejestracja informacji na temat zamówień
  + Dodawanie, usuwanie, aktualizacja zamówień złożonych na produkty
* Rejestracja informacji na samochodów
  + Rejestracja informacji na temat samochodów, które rozwożą produkty wytworzone w piekarni do sklepów

Niefunkcjonalne:

* Integralność
  + Zapewnienie spójności danych poprzez odpowiednie ograniczenia integralności referencyjnej i walidację danych.
  + Unikanie duplikacji danych i zagwarantowanie poprawności powiązań między encjami.
* Użyteczność
  + Przejrzyste prezentowanie informacji, w tym zrozumiałe etykiety i komunikaty dla użytkowników.
* Wydajność
  + Szybkie wyszukiwanie danych na podstawie różnych kryteriów
* Skalowalność
  + Możliwość rozbudowy bazy danych
  + Efektywne zarządzanie dużą ilością danych bez utraty wydajności

Model pojęciowy (chen)



# Wykaz encji logicznych

|  |  |  |
| --- | --- | --- |
| Lp. | Encja-Logiczna | Opis |
| 1 | Sklepy(Klienci) | Encja opisująca sklepy, do których piekarnia dostarcza pieczywo |
| 2 | Zwroty | Encja przechowywująca informacje na temat niesprzedanych produktów w sklepach |
| 3 | SzczegolyZwrotow | Encja zawierająca szczegółowe informacje na temat zwrotów ( np. zwracany produkt czy ilość) |
| 4 | Pracownicy | Encja opisująca pracowników zatrudnionych w piekarni |
| 5 | Stanowiska | Encja przechowywująca informacje na temat stanowisk pracowników w piekarni |
| 6 | ZapisaniNaSzkolenia | Encja przechowywująca informacje na temat pracowników zapisanych na szkolenia ( pracownik oraz informacje na jakie szkolenie został zapisany) |
| 7 | Szkolenia | Encja opisująca szkolenia w piekarni |
| 8 | PrzebiegDniaPracy | Encja zawierająca informacje na temat przebiegu dnia pracy pracowników ( liczba godzin jakie pracownik przepracował w danym dniu) |
| 9 | StawkiPracownikow | Encja zawierająca informacje na temat stawek pracowników |
| 10 | Dostawcy | Encja opisująca dostawców, którzy zaopatrują piekarnie w składniki |
| 11 | Dostawy | Encja przechowywująca informacje na temat dostaw do piekarni ( dostawy są realizowane przed dostawców) |
| 12 | SzczegolyDostawy | Encja przechowywująca informacje na temat składników zamówionych u dostawców przez piekarnie |
| 13 | Magazyn | Encja zawierająca informacje na temat magazynu, do którego dostarczane są składniki przez dostawców |
| 14 | Składniki | Encja przechowywująca informacje na temat składników |
| 15 | Produkty | Encja zawierająca informacje o produktach tworzonych w piekarni |
| 16 | KategorieProduktow | Encja przechowująca informacje na temat kategorii produktów piekarni |
| 17 | Opakowania | Encja opisująca rodzaje opakowań używanych w piekarni |
| 18 | Zamowienia | Encja przechowywująca informacje na temat zamówień produktów z piekarni do sklepów |
| 19 | SzczegolyZamowienia | Encja opisująca szczegóły zamówień produktów z piekarni ( produkty oraz ich ilość) |
| 20 | Województwa | Encja przechowująca województwa |
| 21 | Plcie | Encja przechowująca płcie |
| 22 | Samochody | Encja zawierająca informacje na temat samochodów rozwożących produkty z piekarni |
| 23 | Informacje | Encja przechowywująca informacje na temat powiadomień generowanych przez system |

# Wykaz atrybutów

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sklepy | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdSklepu | LC | NOT-NULL | Atrybut przechowywujący identyfikator sklepu:  Samo inkrementacja:  (1,1) |
| 2 |  | NazwaSklepu | T(50) | NOT-NULL | Atrybut przechowywujący  Nazwę sklepu |
| 3 |  | Adres | T(50) | NOT-NULL | Atrybut przechowywujący  Adres sklepu |
| 4 | AK | Nip | T(10) | NOT-NULL | Atrybut przechowywujący  Numer Nip sklepu |
| 5 |  | Telefon | T(12) | NULL | Atrybut przechowywujący  Numer telefonu stacjonarnego sklepu |
| 6 |  | TelefonKomorkowy | T(14) | NULL | Atrybut przechowywujący  Numer telefonu komórkowego sklepu |
| 7 |  | Email | T(50) | NULL | Atrybut przechowywujący  email sklepu |
| 8 |  | KodPocztowy | T(6) | NOT-NULL | Atrybut przechowywujący kod pocztowy sklepu |
| 9 |  | Miejscowosc | T(35) | NOT-NULL | Atrybut przechowywujący miejscowość sklepu |
| 10 | FK | IdWojewodztwa | LC | NOT-NULL | Atrybut przechowywujący identyfikator województwa |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Zwroty | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdZwrotu | LC | NOT-NULL | Atrybut przechowywujący identyfikator zwrotu:  Samo inkrementacja:  (1,1) |
| 2 | FK | IdSklepu | LC | NOT-NULL | Atrybut przechowywujący identyfikator sklepu |
| 3 |  | DataZwrotu | DATE | NOT-NULL | Atrybut przechowywujący informacje na temat daty zwrotu |
| 4 |  | Powod | T(50) | NOT-NULL | Atrybut opisujący powód zwrotu |
| 5 | FK | IdPracownika | LC | NOT-NULL | Atrybut zawierający informacje na temat identyfikatora pracownika odpowiadającego za zwrot |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SzczegolyZwrotow | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdSzczegZwrotu | LC | NOT-NULL | Atrybut przechowywujący identyfikator szczegółów zwrotu  Samo inkrementacja:  (1,1) |
| 2 | FK | IdZwrotu | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora zwrotu |
| 3 | FK | IdProduktu | LC | NOT-NULL | Atrybuty przechowywujący informacje na temat identyfikatora produktu |
| 4 | FK | IdZamowienia | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora zamówienia |
| 5 |  | Ilosc | T(4) | NOT-NULL | Atrybut przechowywujący informacje na temat ilości zwróconych produktów |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| StawkiPracownikow | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdStawki | LC | NOT-NULL | Atrybut przechowywujący identyfikator stawki:  Samo inkrementacja:  (1,1) |
| 2 |  | KwotaStawki | T(5) | NOT-NULL | Atrybut przechowywujący identyfikator stawki |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pracownicy | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdPracownika | LC | NOT-NULL | Atrybut przechowywujący identyfikator pracownika:  Samo inkrementacja:  (1,1) |
| 2 |  | Imie | T(15) | NOT-NULL | Atrybut przechowywujący imię pracownika |
| 3 |  | Nazwisko | T(30) | NOT-NULL | Atrybut przechowywujący  Nazwisko pracownika |
| 4 | AK | Pesel | T(11) | NOT-NULL | Atrybut przechowywujący pesel pracownika piekarni |
| 5 |  | Telefon | T(12) | NULL | Atrybut przechowywujący  Numer telefonu stacjonarnego pracownika |
| 6 |  | TelefonKomorkowy | T(14) | NULL | Atrybut przechowywujący numer telefonu komórkowego pracownika |
| 7 | FK | IdStanowiska | LC | NOT-NULL | Atrybut przechowywujący identyfikator stanowiska pracownika |
| 8 |  | Email | T(50) | NULL | Atrybut przechowywujący email pracownika |
| 9 | FK | IdStawki | LC | NOT-NULL | Atrybut przechowywujący identyfikator stawki |
| 10 | FK | IdPlec | LC | NOT-NULL | Atrybut przechowujący identyfikator płci pracownika |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Stanowiska | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdStanowiska | LC | NOT-NULL | Atrybut przechowywujący identyfikator stanowiska:  Samo inkrementacja:  (1,1) |
| 2 |  | Stanowisko | T(50) | NOT-NULL | Atrybut przechowywujący nazwę stanowiska |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ZapisaniNaSzkolenia | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdZapisani | LC | NOT-NULL | Atrybut przechowywujący identyfikator zapisanych na szkolenia:  Samo inkrementacja:  (1,1) |
| 2 | FK | IdPracownika | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora pracownika |
| 3 | FK | IdSzkolenia | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora szkolenia |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Szkolenia | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdSzkolenia | LC | NOT-NULL | Atrybut przechowywujący identyfikator szkolenia:  Samo inkrementacja:  (1,1) |
| 2 |  | NazwaSzkolenia | T(50) | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora nazwy szkolenia |
| 3 |  | DataSzkolenia | DATE | NOT-NULL | Atrybut przechowywujący informacje na temat daty szkolenia |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PrzebiegDniaPracy | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdPrzebiegu | LC | NOT-NULL | Atrybut przechowywujący identyfikator przebiegu dnia pracy:  Samo inkrementacja:  (1,1) |
| 2 | FK | IdPracownika | LC | NOT-NULL | Atrybut zawierający informacje na temat identyfikatora pracownika |
| 3 |  | Data | LC | NOT-NULL | Atrybut zawierający informacje na temat daty przebiegu dnia pracy |
| 4 |  | LiczbaGodzin | T(2) | NOT-NULL | Atrybut przechowywujący liczbę przepracowanych godzin pracownika danego dnia pracy |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dostawcy | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdDostawcy | LC | NOT-NULL | Atrybut przechowywujący identyfikator dostawcy:  Samo inkrementacja:  (1,1) |
| 2 |  | NazwaDostawcy | T(50) | NOT-NULL | Atrybut przechowywujący  Nazwę dostawcy |
| 3 |  | Adres | T(50) | NOT-NULL | Atrybut przechowywujący  Adres dostawcy |
| 4 | AK | Nip | T(10) | NOT-NULL | Atrybut przechowywujący  Nip dostawcy |
| 5 |  | Telefon | T(12) | NULL | Atrybut przechowywujący  Numer telefonu stacjonarnego dostawcy |
| 6 |  | TelefonKomorkowy | T(14) | NULL | Atrybut przechowywujący  Numer telefonu komórkowego dostawcy |
| 7 |  | Email | T(50) | NULL | Atrybut przechowywujący  Email dostawcy |
| 8 |  | KodPocztowy | T(6) | NOT-NULL | Atrybut przechowywujący kod pocztowy dostawcy |
| 9 |  | Miejscowosc | T(35) | NOT-NULL | Atrybut przechowywujący miejscowość dostawcy |
| 10 | FK | IdWojewodztwa | LC | NOT-NULL | Atrybut przechowywujący identyfikator województwa |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dostawy | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdDostawy | LC | NOT-NULL | Atrybut przechowywujący identyfikator dostawy:  Samo inkrementacja:  (1,1) |
| 2 | FK | IdDostawcy | LC | NOT-NULL | Atrybut przechowywujący identyfikator dostawcy |
| 3 |  | DataDostawy | DATE | NOT-NULL | Atrybut zawierający informacje na temat daty dostawy |
| 4 |  | KwotaDostawy | T(6) | NOT-NULL | Atrybut opisujący kwotę dostawy (w zł) |
| 5 | FK | IdMagazynu | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora magazynu |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SzczegolyDostaw | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdSzczegoluDostawy | LC | NOT-NULL | Atrybut przechowywujący identyfikator sklepu:  Samo inkrementacja:  (1,1) |
| 2 | FK | IdDostawy | LC | NOT-NULL | Atrybut przechowywujący identyfikator dostawy |
| 3 | FK | IdSkladnika | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora składnika |
| 4 |  | Ilosc | T(6) | NOT-NULL | Atrybut opisujący ilość dostarczanego produktu |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Magazyn | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdMagazynu | LC | NOT-NULL | Atrybut przechowywujący identyfikator magazynu:  Samo inkrementacja:  (1,1) |
| 2 |  | NazwaMagazynu | T(50) | NOT-NULL | Atrybut zawierający informacje na temat nazwy magazynu |
| 3 |  | Adres | T(50) | NOT-NULL | Atrybut przechowywujący adres magazynu |
| 4 |  | KodPocztowy | T(6) | NOT-NULL | Atrybut przechowywujący kod pocztowy magazynu |
| 5 |  | Miejscowosc | T(35) | NOT-NULL | Atrybut przechowywujący miejscowość magazynu |
| 6 | FK | IdWojewodztwa | LC | NOT-NULL | Atrybut przechowywujący identyfikator województwa |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Skladniki | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdSkładnika | LC | NOT-NULL | Atrybut przechowywujący identyfikator składnika:  Samo inkrementacja:  (1,1) |
| 2 |  | NazwaSkladnika | T(30) | NOT-NULL | Atrybut zawierający informacje na temat nazwy składnika |
| 3 |  | OpisSkladnika | T(50) | NULL | Atrybut przechowywujący opis dotyczący składnika |
| 4 |  | JednostkaMiary | T(5) | NOT-NULL | Atrybut opisujący jednostkę miary danego składnika |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Produkty | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdProduktu | LC | NOT-NULL | Atrybut przechowywujący identyfikator produktu:  Samo inkrementacja:  (1,1) |
| 2 | FK | IdKategorii | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora kategorii |
| 3 | FK | IdOpakowania | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora opakowania |
| 4 |  | NazwaProduktu | T(30) | NOT-NULL | Atrybut zawierający nazwę produktu |
| 5 |  | OpisProduktu | T(50) | NOT-NULL | Atrybut zawierający nazwę produktu |
| 6 |  | CenaProduktu | T(4) | NOT-NULL | Atrybut opisujący cenę danego produktu |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| KategorieProdutkow | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdKategorii | LC | NOT-NULL | Atrybut przechowywujący identyfikator kategorii:  Samo inkrementacja:  (1,1) |
| 2 |  | NazwaKategorii | T(30) | NOT-NULL | Atrybut zawierający informacje na temat nazwy kategorii |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Opakowania | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdOpakowania | LC | NOT-NULL | Atrybut przechowywujący identyfikator opakowania:  Samo inkrementacja:  (1,1) |
| 2 |  | RodzajOpakowania | T(30) | NOT-NULL | Atrybut przechowywujący informacje na temat rodzaju opakowania |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Zamowienia | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdZamowienia | LC | NOT-NULL | Atrybut przechowywujący identyfikator zamówienia:  Samo inkrementacja:  (1,1) |
| 2 | FK | IdSklepu | LC | NOT-NULL | Atrybut zawierający informacje na temat identyfikatora sklepu |
| 3 |  | DataZamowienia | DATE | NOT-NULL | Atrybut zawierający informacje na temat daty zamówienia |
| 4 | FK | IdSamochodu | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora samochodu |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SzczegolyZamowien | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdSzczegoluZam | LC | NOT-NULL | Atrybut przechowywujący identyfikator szczegółu zamówienia:  Samo inkrementacja:  (1,1) |
| 2 | FK | IdZamowienia | LC | NOT-NULL | Atrybut zawierający identyfikator zamówienia |
| 3 | FK | IdProduktu | LC | NOT-NULL | Atrybut zawierający identyfikator produktu |
| 4 |  | Ilosc | T(4) | NOT-NULL | Atrybut opisujący ilość zamówionego produktu (szt) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Plcie | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdPlec | LC | NOT-NULL | Atrybut przechowywujący identyfikator płci pracownika |
| 2 |  | Plec | T(9) | NOT-NULL | Atrybut przechowywujący płeć pracownika |
| 3 |  | SymbolPlci | T(1) | NOT-NULL | Atrybut przechowywujący symbol płci pracownika |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Wojewodztwa | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdWojewodztwa | LC | NOT-NULL | Atrybut przechowywujący identyfikator województwa |
| 2 |  | Wojewodztwo | T(19) | NOT-NULL | Atrybut przechowywujący nazwę województwa |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Samochody | | | | | |
| Lp. | Atrybut kluczowy | Atrybut | Typ danych | Czy wymagany? | Opis |
| 1 | PK | IdSamochodu | LC | NOT-NULL | Atrybut przechowywujący identyfikator sklepu:  Samo inkrementacja:  (1,1) |
| 2 | AK | NumerRejestracyjny | T(8) | NOT-NULL | Atrybut opisujący numer rejestracyjny pojazdu |
| 3 | AK | Vin | T(17) | NOT-NULL | Dane na temat numeru VIN samochodu |
| 4 |  | RodzajPaliwa | T(2) | NOT-NULL | Atrybut zawierający informacje na temat typu paliwa |
| 5 |  | Marka | T(30) | NOT-NULL | Atrybut zawierający informacje o marce pojazdu |
| 6 | FK | IdPracownika | LC | NOT-NULL | Atrybut przechowywujący informacje na temat identyfikatora pracownika |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Informacje | | | | | |
| 1 | PK | IdInfo | LC | NOT-NULL | Atrybut przechowywujący identyfikator informacji: Samo inkrementacja(1,1) |
| 2 |  | Wiadomosc | T(500) | NOT-NULL | Atrybut przechowywujący informacje na temat zapisanego komunikatu |
| 3 |  | Gdzie | T(128) | NOT-NULL | Atrybut zawierający informacje na temat gdzie wystąpił dany komunikat |
| 4 |  | Data | DATETIME | NOT-NULL | Atrybut przechowywujący datę wystąpienia komunikatu, wartość domyślna: DEFAULT CURRENT\_TIMESTAMP |

# Wykaz związków

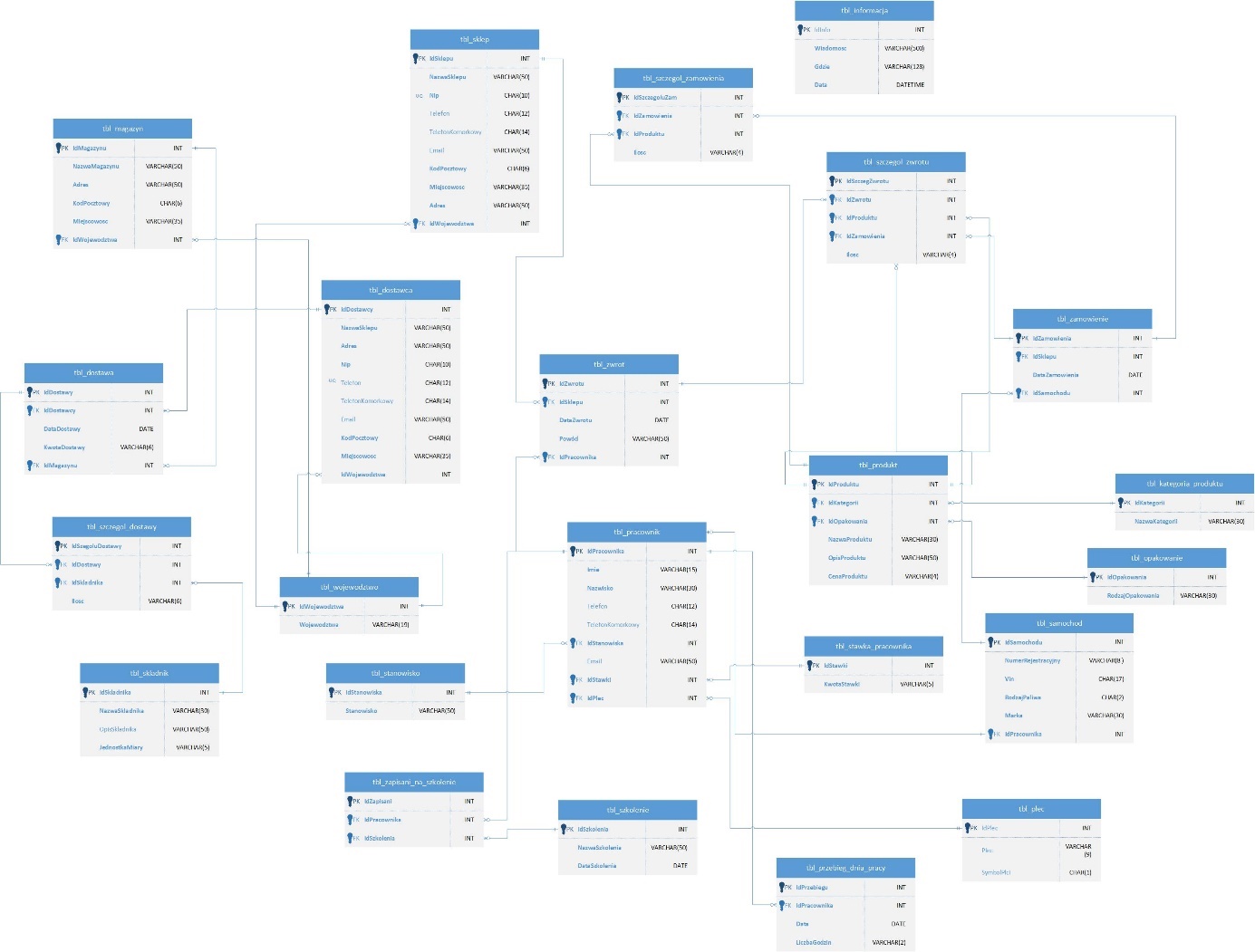
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lp | Encja 1 | Encja 2 | Typ | Opcje związku | Kardynalność | Opis |
| 1 | Wojewodztwo | Sklepy | 1:N | Wymagalny | Jeden do wielu | Województwo jest przypisane do Sklepu, Sklep ma przypisane Województwo |
| 2 | Sklepy | Zwroty | 1:N | Wymagalny | Jeden do wielu | Sklep posiada Zwroty, Zwroty są przypisane do Sklepu |
| 3 | Zwroty | SzczegolyZwrotow | 1:N | Wymagalny | Jeden do wielu | Zwroty posiadają SzczegolyZwrotow, SzczegolyZwrotow są przypisane do Zwroty |
| 4 | Produkty | SzczegolyZwrotow | 1:N | Wymagalny | Jeden do wielu | Produkty są przypisane do SzczegolyZwrotow, Szczegóły zwrotu posiadają Produkty |
| 5 | Zamówienia | SzczegolyZwrotow | 1:N | Wymagalny | Jeden do wielu | Zamowienia posiadają SzczegolyZwrotów, SzczegolyZwrotów są przypisane do Zamówienia |
| 6 | StawkiPracownika | Pracownicy | 1:N | Wymagalny | Jeden do wielu | StawkiPracownika są przypisane do Pracownicy, Pracownicy posiadają StawkiPracownika |
| 7 | Stanowiska | Pracownicy | 1:N | Wymagalny | Jeden do wielu | Stanowiska są przypisane do Pracownicy, Pracownicy posiadają Stanowiska |
| 8 | Pracownicy | ZapisaniNaSzkolenia | 1:N | Wymagalny | Jeden do wielu | Pracownicy są przypisani do ZapisaniNaSzkolenia, ZapisaniNaSzkolenia posiadają Pracowników |
| 9 | Szkolenia | ZapisaniNaSzkolenia | 1:N | Wymagalny | Jeden do wielu | Szkolenia posiadają ZapisaniNaSzkolenia, ZapisaniNaSzkolenia są przypisani do Szkolenia |
| 10 | Pracownicy | PrzebiegDniaPracy | 1:N | Wymagalny | Jeden do wielu | Pracownik posiada PrzebiegDniaPracy, PrzebiegDniaPracy jest przypisany do Pracownika |
| 11 | Wojewodztwo | Dostawcy | 1:N | Wymagalny | Jeden do wielu | Województwo jest przypisane do Dostawcy, Dostawcy posiada Województwo |
| 12 | Dostawcy | Dostawy | 1:N | Wymagalny | Jeden do wielu | Dostawcy są przypisani do dostaw, dostawy posiadają dostawców |
| 13 | Dostawy | SzczegolyDostaw | 1:N | Wymagalny | Jeden do wielu | Dostawy posiadają SzczegółyDostaw, SzczegółyDostaw są przypisani do Dostawy |
| 14 | Magazyny | Dostawy | 1:N | Opcjonalny | Jeden do wielu | Magazyny są przypisani do Dostawy, Dostawy posiadają magazyny |
| 15 | Składniki | SzczegolyDostaw | 1:N | Wymagalny | Jeden do wielu | Składniki są przypisani do SzczegółyDostaw, SzczegółyDostaw posiadają składniki |
| 16 | Województwo | Magazyn | 1:N | Wymagalny | Jeden do wielu | Województwo jest przypisane do Sklepu, Sklep ma Województwo |
| 17 | KategorieProduktow | Produkty | 1:N | Wymagalny | Jeden do wielu | KategorieProduktów są przypisane do Produkty, Produkty posiadają KategorieProduktów |
| 18 | Opakowanie | Produkty | 1:N | Wymagalny | Jeden do jednego | Opakowanie są przypisane do Produkt, Produkt posiada Opakowanie |
| 19 | Zamowienia | SzczegolyZamowienia | 1:N | Wymagalny | Jeden do wielu | Zamówienie posiada SzczegółyZamówienia, SzczegółyZamówienia przypisane są do Zamówienia |
| 20 | Produkty | SzczegolyZamowienia | 1:N | Wymagalny | Jeden do wielu | Produkty przypisane są do SzczegółyZamówienia, SzczegółyZamówienia posiadają produkty |
| 21 | Płcie | Pracownicy | 1:N | Wymagalny | Jeden do wielu | Płcie przypisane są do Pracowników, Pracownicy posiadają płeć |
| 22 | Pracownicy | Samochody | 1:1 | Wymagalny | Jeden do jednego | Pracownicy przypisane są do Samochody, Samochody posiadają Pracownicy |
| 23 | Samochody | Zamowienia | 1:N | Wymagalny | Jeden do wielu | Samochody rozwożą zamówienia, zamówienia są przypisane do samochodów |
| 24 | Pracownik | Zwroty | 1:N | Wymagalny | Jeden do wielu | Pracownik przyjmuje zwroty, zwroty są przyjmowane przez pracownika |

# Model logiczny

Obraz zawierający tekst, diagram, Równolegle, zrzut ekranu

Opis wygenerowany automatycznie

Model fizyczny



# Wykaz tabel fizycznych

- tbl\_sklep

-tbl\_zwrot

-tbl\_szczeg\_zwrotu

-tbl\_pracownik

-tbl\_stanowisko

-tbl\_zapisani\_na\_szkolenie

-tbl\_szkolenie

-tbl\_przebieg\_dnia\_pracy

-tbl\_stawka\_pracownika

-tbl\_dostawca

-tbl\_dostawa

-tbl\_szczeg\_dostawy

-tbl\_magazyn

-tbl\_skladnik

-tbl\_produkt

-tbl\_kategoria\_produktu

-tbl\_opakowanie

-tbl\_zamowienie

-tbl\_szczeg\_zamowienia

-tbl\_wojewodztwo

-tbl\_plec

-tbl\_samochod

-tbl\_informacja

# Skrypty SQL tworzące strukturę tabel i powiązania(MS SERVER)

CREATE DATABASE Piekarnia;

USE Piekarnia

GO

CREATE TABLE Wojewodztwa(

IdWojewodztwa INT PRIMARY KEY NOT NULL,

Wojewodztwo VARCHAR(19) NOT NULL

);

CREATE TABLE Sklepy (

IdSklepu INT PRIMARY KEY NOT NULL,

NazwaSklepu VARCHAR(50) NOT NULL,

Adres VARCHAR(50) NOT NULL,

Nip VARCHAR(10) NOT NULL UNIQUE,

Telefon VARCHAR(12),

TelefonKomorkowy VARCHAR(14),

Email VARCHAR(50),

KodPocztowy VARCHAR(6) NOT NULL,

Miejscowosc VARCHAR(35) NOT NULL,

IdWojewodztwa INT NOT NULL,

FOREIGN KEY (IdWojewodztwa) REFERENCES Wojewodztwa(IdWojewodztwa)

);

CREATE TABLE KategorieProduktow (

IdKategorii INT PRIMARY KEY NOT NULL,

NazwaKategorii VARCHAR(30) NOT NULL

);

CREATE TABLE Opakowania (

IdOpakowania INT PRIMARY KEY NOT NULL,

RodzajOpakowania VARCHAR(30) NOT NULL

);

CREATE TABLE Produkty (

IdProduktu INT PRIMARY KEY NOT NULL,

IdKategorii INT NOT NULL,

IdOpakowania INT NOT NULL,

NazwaProduktu VARCHAR(30) NOT NULL,

OpisProduktu VARCHAR(50) NOT NULL,

CenaProduktu DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (IdKategorii) REFERENCES KategorieProduktow(IdKategorii),

FOREIGN KEY (IdOpakowania) REFERENCES Opakowania(IdOpakowania)

);

CREATE TABLE Plcie (

IdPlec INT PRIMARY KEY NOT NULL,

Plec VARCHAR(9) NOT NULL,

SymbolPlci CHAR(1) NOT NULL

);

CREATE TABLE Stanowiska (

IdStanowiska INT PRIMARY KEY NOT NULL,

Stanowisko VARCHAR(50) NOT NULL

);

CREATE TABLE StawkiPracownikow (

IdStawki INT PRIMARY KEY NOT NULL,

KwotaStawki DECIMAL(10, 2) NOT NULL

);

CREATE TABLE Pracownicy (

IdPracownika INT PRIMARY KEY NOT NULL,

Imie VARCHAR(15) NOT NULL,

Nazwisko VARCHAR(30) NOT NULL,

Pesel VARCHAR(11) NOT NULL UNIQUE,

Telefon VARCHAR(12),

TelefonKomorkowy VARCHAR(14),

IdStanowiska INT NOT NULL,

Email VARCHAR(50),

IdStawki INT NOT NULL,

IdPlec INT NOT NULL

FOREIGN KEY (IdStanowiska) REFERENCES Stanowiska(IdStanowiska),

FOREIGN KEY (IdStawki) REFERENCES StawkiPracownikow(IdStawki),

FOREIGN KEY (IdPlec) REFERENCES Plcie(IdPlec)

);

CREATE TABLE Szkolenia (

IdSzkolenia INT PRIMARY KEY NOT NULL,

NazwaSzkolenia VARCHAR(50) NOT NULL,

DataSzkolenia DATE NOT NULL

);

CREATE TABLE ZapisaniNaSzkolenia (

IdZapisani INT PRIMARY KEY NOT NULL,

IdPracownika INT NOT NULL,

IdSzkolenia INT NOT NULL,

FOREIGN KEY (IdPracownika) REFERENCES Pracownicy(IdPracownika),

FOREIGN KEY (IdSzkolenia) REFERENCES Szkolenia(IdSzkolenia)

);

CREATE TABLE PrzebiegDniaPracy (

IdPrzebiegu INT PRIMARY KEY NOT NULL,

IdPracownika INT NOT NULL,

Data DATE NOT NULL,

LiczbaGodzin INT NOT NULL,

FOREIGN KEY (IdPracownika) REFERENCES Pracownicy(IdPracownika)

);

CREATE TABLE Zamowienia (

IdZamowienia INT PRIMARY KEY NOT NULL,

IdSklepu INT NOT NULL,

DataZamowienia DATE NOT NULL,

IdSamochodu INT NOT NULL,

FOREIGN KEY (IdSklepu) REFERENCES Sklepy(IdSklepu),

FOREIGN KEY (IdSamochodu) REFERENCES Samochody (IdSamochodu)

);

CREATE TABLE SzczegolyZamowien (

IdSzczegoluZam INT PRIMARY KEY NOT NULL,

IdZamowienia INT NOT NULL,

IdProduktu INT NOT NULL,

Ilosc INT NOT NULL,

FOREIGN KEY (IdZamowienia) REFERENCES Zamowienia(IdZamowienia),

FOREIGN KEY (IdProduktu) REFERENCES Produkty(IdProduktu)

);

CREATE TABLE Samochody (

IdSamochodu INT PRIMARY KEY NOT NULL,

NumerRejestracyjny VARCHAR(8) NOT NULL UNIQUE,

Vin VARCHAR(17) NOT NULL UNIQUE,

RodzajPaliwa CHAR(2) NOT NULL,

Marka VARCHAR(30) NOT NULL,

IdPracownika INT NOT NULL,

FOREIGN KEY (IdPracownika) REFERENCES Pracownicy(IdPracownika)

);

CREATE TABLE Zwroty (

IdZwrotu INT PRIMARY KEY NOT NULL,

IdSklepu INT NOT NULL,

DataZwrotu DATE NOT NULL,

Powód VARCHAR(50) NOT NULL,

IdPracownika INT NOT NULL,

FOREIGN KEY (IdSklepu) REFERENCES Sklepy(IdSklepu),

FOREIGN KEY (IdPracownika) REFERENCES Pracownicy(IdPracownika)

);

CREATE TABLE SzczegolyZwrotow (

IdSzczegZwrotu INT PRIMARY KEY NOT NULL,

IdZwrotu INT NOT NULL,

IdProduktu INT NOT NULL,

IdZamowienia INT NOT NULL,

Ilosc INT NOT NULL,

FOREIGN KEY (IdZwrotu) REFERENCES Zwroty(IdZwrotu),

FOREIGN KEY (IdProduktu) REFERENCES Produkty(IdProduktu),

FOREIGN KEY (IdZamowienia) REFERENCES Zamowienia(IdZamowienia)

);

CREATE TABLE Magazyn (

IdMagazynu INT PRIMARY KEY NOT NULL,

NazwaMagazynu VARCHAR(50) NOT NULL,

Adres VARCHAR(50) NOT NULL,

KodPocztowy VARCHAR(6) NOT NULL,

Miejscowosc VARCHAR(35) NOT NULL,

IdWojewodztwa INT NOT NULL,

FOREIGN KEY (IdWojewodztwa) REFERENCES Wojewodztwa(IdWojewodztwa)

);

CREATE TABLE Skladniki (

IdSkładnika INT PRIMARY KEY NOT NULL,

NazwaSkladnika VARCHAR(30) NOT NULL,

OpisSkladnika VARCHAR(50),

JednostkaMiary VARCHAR(5) NOT NULL

);

CREATE TABLE Dostawcy (

IdDostawcy INT PRIMARY KEY NOT NULL,

NazwaDostawcy VARCHAR(50) NOT NULL,

Adres VARCHAR(50) NOT NULL,

Nip VARCHAR(10) NOT NULL UNIQUE,

Telefon VARCHAR(12),

TelefonKomorkowy VARCHAR(14),

Email VARCHAR(50),

KodPocztowy VARCHAR(6) NOT NULL,

Miejscowosc VARCHAR(35) NOT NULL,

IdWojewodztwa INT NOT NULL,

FOREIGN KEY (IdWojewodztwa) REFERENCES Wojewodztwa(IdWojewodztwa)

);

CREATE TABLE Dostawy (

IdDostawy INT PRIMARY KEY NOT NULL,

IdDostawcy INT NOT NULL,

DataDostawy DATE NOT NULL,

KwotaDostawy DECIMAL(10, 2) NOT NULL,

IdMagazynu INT NOT NULL,

FOREIGN KEY (IdDostawcy) REFERENCES Dostawcy(IdDostawcy),

FOREIGN KEY (IdMagazynu) REFERENCES Magazyn(IdMagazynu)

);

CREATE TABLE SzczegolyDostaw (

IdSzczegoluDostawy INT PRIMARY KEY NOT NULL,

IdDostawy INT NOT NULL,

IdSkladnika INT NOT NULL,

Ilosc INT NOT NULL,

FOREIGN KEY (IdDostawy) REFERENCES Dostawy(IdDostawy),

FOREIGN KEY (IdSkladnika) REFERENCES Skladniki(IdSkładnika)

);

# Wykaz indeksów

-idx\_nazwa\_sklepu

-ui\_nip\_sklepu

-ui\_nip\_dostawcy

-ui\_pesel

-idx\_data\_dzien\_rob

-idx\_nazwa\_skladnika

-idx\_nazwa\_produktu

-idx\_data\_zamowienia

-ui\_numer\_rej

-ui\_vin

# Skrypty tworzące indeksy

-- Indeks dla kolumny NazwaSklepu w tabeli Sklepy

CREATE INDEX idx\_nazwa\_sklepu ON Sklepy (NazwaSklepu);

-- Unikalny indeks dla kolumny Nip w tabeli Sklepy

CREATE UNIQUE INDEX ui\_nip\_sklepu ON Sklepy (Nip);

-- Unikalny indeks dla kolumny Nip w tabeli Dostawcy

CREATE UNIQUE INDEX ui\_nip\_dostawcy ON Dostawcy (Nip);

-- Unikalny indeks dla kolumny Pesel w tabeli Pracownicy

CREATE UNIQUE INDEX ui\_pesel ON Pracownicy (Pesel);

-- Indeks dla kolumny Data w tabeli PrzebiegDniaPracy

CREATE INDEX idx\_data\_dzien\_rob ON PrzebiegDniaPracy (Data);

-- Indeks dla kolumny NazwaSkladnika w tabeli Skladniki

CREATE INDEX idx\_nazwa\_skladnika ON Skladniki (NazwaSkladnika);

-- Indeks dla kolumny NazwaProduktu w tabeli Produkty

CREATE INDEX idx\_nazwa\_produktu ON Produkty (NazwaProduktu);

-- Indeks dla kolumny DataZamowienia w tabeli Zamowienia

CREATE INDEX idx\_data\_zamowienia ON Zamowienia (DataZamowienia);

-- Unikalny indeks dla kolumny NumerRejestracyjny w tabeli Samochody

CREATE UNIQUE INDEX ui\_numer\_rej ON Samochody (NumerRejestracyjny);

-- Unikalny indeks dla kolumny Vin w tabeli Samochody

CREATE UNIQUE INDEX ui\_vin ON Samochody (Vin);

# Wykaz widoków

-vw\_PracownicySzkolenia

-vw\_ProduktyKategorie

-vw\_ZamowieniaSzczegoly

-vw\_DostawcyDostawy

-vw\_SklepyZwroty

-vw\_ProduktyOpakowania

-vw\_PracownicySamochody

-vw\_DostawyMagazyny

-vw\_SkładnikiDostawy

-vw\_DostawySzczegoly

-vw\_MagazynySkladniki

# Skrypty tworzące widoki

--Widoki

--Widok "vw\_PracownicySzkolenia"

GO

CREATE VIEW vw\_PracownicySzkolenia AS

SELECT

P.IdPracownika,

P.Imie,

P.Nazwisko,

P.Pesel,

P.Telefon,

P.TelefonKomorkowy,

P.Email,

S.Stanowisko,

St.KwotaStawki,

Pl.Plec,

Pl.SymbolPlci,

Sz.IdSzkolenia,

Sz.NazwaSzkolenia,

Sz.DataSzkolenia

FROM

Pracownicy P

INNER JOIN Stanowiska S ON P.IdStanowiska = S.IdStanowiska

INNER JOIN StawkiPracownikow St ON P.IdStawki = St.IdStawki

INNER JOIN Plcie Pl ON P.IdPlec = Pl.IdPlec

INNER JOIN ZapisaniNaSzkolenia ZS ON P.IdPracownika = ZS.IdPracownika

INNER JOIN Szkolenia Sz ON ZS.IdSzkolenia = Sz.IdSzkolenia;

GO

--Widok "vw\_ProduktyKategorie"

CREATE VIEW vw\_ProduktyKategorie AS

SELECT

P.IdProduktu,

P.NazwaProduktu,

P.OpisProduktu,

P.CenaProduktu,

K.IdKategorii,

K.NazwaKategorii

FROM

Produkty P

INNER JOIN KategorieProduktow K ON P.IdKategorii = K.IdKategorii;

GO

--Widok "vw\_ZamowieniaSzczegoly"

CREATE VIEW vw\_ZamowieniaSzczegoly AS

SELECT

Z.IdZamowienia,

Z.DataZamowienia,

Z.DataZamowienia,

S.IdSklepu,

S.NazwaSklepu,

S.Adres AS AdresSklepu,

SZ.IdSzczegoluZam,

P.IdProduktu,

P.NazwaProduktu,

P.OpisProduktu,

P.CenaProduktu,

SZ.Ilosc

FROM

Zamowienia Z

INNER JOIN SzczegolyZamowien SZ ON Z.IdZamowienia = SZ.IdZamowienia

INNER JOIN Produkty P ON SZ.IdProduktu = P.IdProduktu

INNER JOIN Sklepy S ON Z.IdSklepu = S.IdSklepu;

GO

--Widok "vw\_DostawcyDostawy"

CREATE VIEW vw\_DostawcyDostawy AS

SELECT

D.IdDostawcy,

D.NazwaDostawcy,

D.Adres AS AdresDostawcy,

D.Nip AS NipDostawcy,

D.Telefon AS TelefonDostawcy,

D.TelefonKomorkowy AS TelefonKomorkowyDostawcy,

D.Email AS EmailDostawcy,

D.KodPocztowy AS KodPocztowyDostawcy,

D.Miejscowosc AS MiejscowoscDostawcy,

W.Wojewodztwo AS WojewodztwoDostawcy,

DS.IdDostawy,

DS.DataDostawy,

DS.KwotaDostawy,

M.NazwaMagazynu,

M.Adres AS AdresMagazynu

FROM

Dostawcy D

INNER JOIN Dostawy DS ON D.IdDostawcy = DS.IdDostawcy

INNER JOIN Magazyn M ON DS.IdMagazynu = M.IdMagazynu

INNER JOIN Wojewodztwa W ON D.IdWojewodztwa = W.IdWojewodztwa;

GO

--Widok "vw\_SklepyZwroty"

CREATE VIEW vw\_SklepyZwroty AS

SELECT

S.IdSklepu,

S.NazwaSklepu,

S.Adres AS AdresSklepu,

S.Nip AS NipSklepu,

S.Telefon AS TelefonSklepu,

S.TelefonKomorkowy AS TelefonKomorkowySklepu,

S.Email AS EmailSklepu,

S.KodPocztowy AS KodPocztowySklepu,

S.Miejscowosc AS MiejscowoscSklepu,

W.Wojewodztwo AS WojewodztwoSklepu,

Z.IdZwrotu,

Z.DataZwrotu,

Z.Powód AS PowodZwrotu,

Z.IdPracownika

FROM

Sklepy S

INNER JOIN Zwroty Z ON S.IdSklepu = Z.IdSklepu

INNER JOIN Wojewodztwa W ON S.IdWojewodztwa = W.IdWojewodztwa;

GO

--Widok "vw\_ProduktyOpakowania"

CREATE VIEW vw\_ProduktyOpakowania AS

SELECT

P.IdProduktu,

P.NazwaProduktu,

P.OpisProduktu,

P.CenaProduktu,

O.IdOpakowania,

O.RodzajOpakowania

FROM

Produkty P

INNER JOIN Opakowania O ON P.IdOpakowania = O.IdOpakowania;

GO

--Widok "vw\_PracownicySamochody"

CREATE VIEW vw\_PracownicySamochody AS

SELECT

P.IdPracownika,

P.Imie,

P.Nazwisko,

P.Pesel,

P.Telefon,

P.TelefonKomorkowy,

P.Email,

S.Stanowisko,

St.KwotaStawki,

Pl.Plec,

Pl.SymbolPlci,

Sa.IdSamochodu,

Sa.NumerRejestracyjny,

Sa.Vin,

Sa.RodzajPaliwa,

Sa.Marka

FROM

Pracownicy P

INNER JOIN Stanowiska S ON P.IdStanowiska = S.IdStanowiska

INNER JOIN StawkiPracownikow St ON P.IdStawki = St.IdStawki

INNER JOIN Plcie Pl ON P.IdPlec = Pl.IdPlec

LEFT JOIN Samochody Sa ON P.IdPracownika = Sa.IdPracownika;

GO

--Widok "vw\_DostawyMagazyny"

CREATE VIEW vw\_DostawyMagazyny AS

SELECT

DS.IdDostawy,

DS.DataDostawy,

DS.KwotaDostawy,

D.IdDostawcy,

D.NazwaDostawcy,

D.Adres AS AdresDostawcy,

D.Nip AS NipDostawcy,

D.Telefon AS TelefonDostawcy,

D.TelefonKomorkowy AS TelefonKomorkowyDostawcy,

D.Email AS EmailDostawcy,

D.KodPocztowy AS KodPocztowyDostawcy,

D.Miejscowosc AS MiejscowoscDostawcy,

W.Wojewodztwo AS WojewodztwoDostawcy,

M.IdMagazynu,

M.NazwaMagazynu,

M.Adres AS AdresMagazynu,

M.KodPocztowy AS KodPocztowyMagazynu,

M.Miejscowosc AS MiejscowoscMagazynu

FROM

Dostawy DS

INNER JOIN Dostawcy D ON DS.IdDostawcy = D.IdDostawcy

INNER JOIN Magazyn M ON DS.IdMagazynu = M.IdMagazynu

INNER JOIN Wojewodztwa W ON D.IdWojewodztwa = W.IdWojewodztwa;

GO

--Widok "vw\_SkładnikiDostawy"

CREATE VIEW vw\_SkladnikiDostawy AS

SELECT

DS.IdDostawy,

DS.DataDostawy,

DS.KwotaDostawy,

DO.IdDostawcy,

DO.NazwaDostawcy,

DO.Adres AS AdresDostawcy,

DO.Nip AS NipDostawcy,

DO.Telefon AS TelefonDostawcy,

DO.TelefonKomorkowy AS TelefonKomorkowyDostawcy,

DO.Email AS EmailDostawcy,

DO.KodPocztowy AS KodPocztowyDostawcy,

DO.Miejscowosc AS MiejscowoscDostawcy,

M.IdMagazynu,

M.NazwaMagazynu,

M.Adres AS AdresMagazynu,

M.KodPocztowy AS KodPocztowyMagazynu,

M.Miejscowosc AS MiejscowoscMagazynu,

SK.IdSkładnika,

SK.NazwaSkladnika,

SK.OpisSkladnika,

SK.JednostkaMiary,

SD.Ilosc

FROM

SzczegolyDostaw SD

INNER JOIN Dostawy DS ON SD.IdDostawy = DS.IdDostawy

INNER JOIN Dostawcy DO ON DS.IdDostawcy = DO.IdDostawcy

INNER JOIN Magazyn M ON DS.IdMagazynu = M.IdMagazynu

INNER JOIN Skladniki SK ON SD.IdSkladnika = SK.IdSkładnika;

GO

--Widok "vw\_DostawySzczegoly"

CREATE VIEW vw\_DostawySzczegoly AS

SELECT

DS.IdDostawy,

DS.DataDostawy,

DS.KwotaDostawy,

D.IdDostawcy,

D.NazwaDostawcy,

D.Adres AS AdresDostawcy,

D.Nip AS NipDostawcy,

D.Telefon AS TelefonDostawcy,

D.TelefonKomorkowy AS TelefonKomorkowyDostawcy,

D.Email AS EmailDostawcy,

D.KodPocztowy AS KodPocztowyDostawcy,

D.Miejscowosc AS MiejscowoscDostawcy,

W.Wojewodztwo AS WojewodztwoDostawcy,

M.IdMagazynu,

M.NazwaMagazynu,

M.Adres AS AdresMagazynu,

M.KodPocztowy AS KodPocztowyMagazynu,

M.Miejscowosc AS MiejscowoscMagazynu,

S.IdSkładnika,

S.NazwaSkladnika,

S.OpisSkladnika,

S.JednostkaMiary,

SD.Ilosc

FROM

SzczegolyDostaw SD

INNER JOIN Dostawy DS ON SD.IdDostawy = DS.IdDostawy

INNER JOIN Dostawcy D ON DS.IdDostawcy = D.IdDostawcy

INNER JOIN Magazyn M ON DS.IdMagazynu = M.IdMagazynu

INNER JOIN Wojewodztwa W ON D.IdWojewodztwa = W.IdWojewodztwa

INNER JOIN Skladniki S ON SD.IdSkladnika = S.IdSkładnika;

GO

--Widok "vw\_MagazynySkladniki"

CREATE VIEW vw\_MagazynySkladniki AS

SELECT

M.IdMagazynu,

M.NazwaMagazynu,

M.Adres AS AdresMagazynu,

M.KodPocztowy AS KodPocztowyMagazynu,

M.Miejscowosc AS MiejscowoscMagazynu,

W.Wojewodztwo AS WojewodztwoMagazynu,

S.IdSkładnika,

S.NazwaSkladnika,

S.OpisSkladnika,

S.JednostkaMiary,

SD.Ilosc,

D.IdDostawy,

D.DataDostawy,

DO.IdDostawcy,

DO.NazwaDostawcy

FROM

Magazyn M

INNER JOIN Wojewodztwa W ON M.IdWojewodztwa = W.IdWojewodztwa

INNER JOIN Dostawy D ON M.IdMagazynu = D.IdMagazynu

INNER JOIN Dostawcy DO ON D.IdDostawcy = DO.IdDostawcy

INNER JOIN SzczegolyDostaw SD ON D.IdDostawy = SD.IdDostawy

INNER JOIN Skladniki S ON SD.IdSkladnika = S.IdSkładnika;

GO

# Skrypty SQL rejestrujące przykładowe dane

INSERT INTO Wojewodztwa (IdWojewodztwa, Wojewodztwo) VALUES

(1, 'dolnośląskie'),

(2, 'kujawsko-pomorskie'),

(3, 'lubelskie'),

(4, 'lubuskie'),

(5, 'łódzkie'),

(6, 'małopolskie'),

(7, 'mazowieckie'),

(8, 'opolskie'),

(9, 'podkarpackie'),

(10, 'podlaskie'),

(11, 'pomorskie'),

(12, 'śląskie'),

(13, 'świętokrzyskie'),

(14, 'warmińsko-mazurskie'),

(15, 'wielkopolskie'),

(16, 'zachodniopomorskie');

INSERT INTO Sklepy (IdSklepu, NazwaSklepu, Adres, Nip, Telefon, TelefonKomorkowy, Email, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES

(1, 'Piekarnia Warszawska', 'Grójecka 7', '1234567890', '111222333', '833199222', 'warszawska@piekarnia.pl', '00-001', 'Warszawa', 6),

(2, 'Piekarnia Mąka', 'Kopcińskiego 64', '1234567891', '123123123', '033823182', 'maka@piekarnia.pl', '60-001', 'Łódź', 5),

(3, 'Piekarnia Pawełek', 'Legionów 66', '1234567892', '456456456', '666777111', 'pawelek@piekarnia.pl', '30-001', 'Wołomin', 6),

(4, 'Piekarnia Putka', 'Piotrkowska 193', '1234567893', '444555666', '166166611', 'putka@piekarnia.pl', '40-001', 'Łódź', 5),

(5, 'Piekarnia Kłos', 'Żelazna 1', '1234567894', '787787787', '957217222', 'klos@piekarnia.pl', '50-001', 'Katowica', 12);

INSERT INTO KategorieProduktow (IdKategorii, NazwaKategorii) VALUES

(1, 'Chleby'),

(2, 'Bułki'),

(3, 'Ciasta'),

(4, 'Przekąski Słone'),

(5, 'Przekąski Słodkie');

INSERT INTO Opakowania (IdOpakowania, RodzajOpakowania) VALUES

(1, 'Papier'),

(2, 'Folia'),

(3, 'Pudełko kartonowe'),

(4, 'Torebka papierowa'),

(5, 'Plastik');

INSERT INTO Produkty (IdProduktu, IdKategorii, IdOpakowania, NazwaProduktu, OpisProduktu, CenaProduktu) VALUES

(1, 1, 1, 'Chleb Żytni', 'Chleb żytni na zakwasie', 4.50),

(2, 2, 4, 'Bułka Kajzerka', 'Tradycyjna bułka kajzerka', 0.60),

(3, 3, 3, 'Ciasto Czekoladowe', 'Ciasto czekoladowe z polewą', 12.00),

(4, 4, 2, 'Bagietka Czosnkowa', 'Bagietka czosnkowa', 3.00),

(5, 5, 2, 'Muffin Czekoladowy', 'Muffin czekoladowy z kawałkami czekolady', 2.50);

-- Wypełnienie tabeli Plcie

INSERT INTO Plcie (IdPlec, Plec, SymbolPlci) VALUES

(1, 'Mężczyzna', 'M'),

(2, 'Kobieta', 'K');

-- Wypełnienie tabeli Stanowiska

INSERT INTO Stanowiska (IdStanowiska, Stanowisko) VALUES

(1, 'Sprzedawca'),

(2, 'Piekarz'),

(3, 'Kierownik Sklepu'),

(4, 'Magazynier'),

(5, 'Kierowca');

-- Wypełnienie tabeli StawkiPracownikow

INSERT INTO StawkiPracownikow (IdStawki, KwotaStawki) VALUES

(1, 15.00),

(2, 18.00),

(3, 20.00),

(4, 22.00),

(5, 25.00);

-- Wypełnienie tabeli Szkolenia

INSERT INTO Szkolenia (IdSzkolenia, NazwaSzkolenia, DataSzkolenia) VALUES

(1, 'Bezpieczeństwo pracy', '2024-06-01'),

(2, 'Obsługa klienta', '2024-06-02'),

(3, 'Techniki sprzedaży', '2024-06-03'),

(4, 'Przygotowanie pieczywa', '2024-06-04'),

(5, 'Zarządzanie zespołem', '2024-06-05');

-- Wypełnienie tabeli Pracownicy

INSERT INTO Pracownicy (IdPracownika, Imie, Nazwisko, Pesel, Telefon, TelefonKomorkowy, IdStanowiska, Email, IdStawki, IdPlec) VALUES

(1, 'Jan', 'Kowalski', '12345678901', '222333444', '666777888', 1, 'jan.kowalski@piekarnia.pl', 1, 1),

(2, 'Anna', 'Nowak', '12345678902', '222333445', '666777889', 2, 'anna.nowak@piekarnia.pl', 2, 2),

(3, 'Piotr', 'Wiśniewski', '12345678903', '222333446', '666777890', 3, 'piotr.wisniewski@piekarnia.pl', 3, 1),

(4, 'Katarzyna', 'Wójcik', '12345678904', '222333447', '666777891', 4, 'katarzyna.wojcik@piekarnia.pl', 4, 2),

(5, 'Tomasz', 'Kowalczyk', '12345678905', '222333448', '666777892', 5, 'tomasz.kowalczyk@piekarnia.pl', 5, 1);

-- Wypełnienie tabeli ZapisaniNaSzkolenia

INSERT INTO ZapisaniNaSzkolenia (IdZapisani, IdPracownika, IdSzkolenia) VALUES

(1, 1, 1),

(2, 2, 2),

(3, 3, 3),

(4, 4, 4),

(5, 5, 5);

-- Wypełnienie tabeli PrzebiegDniaPracy

INSERT INTO PrzebiegDniaPracy (IdPrzebiegu, IdPracownika, Data, LiczbaGodzin) VALUES

(1, 1, '2024-05-01', 8),

(2, 2, '2024-05-01', 8),

(3, 3, '2024-05-01', 8),

(4, 4, '2024-05-01', 8),

(5, 5, '2024-05-01', 8);

INSERT INTO Skladniki (IdSkładnika, NazwaSkladnika, OpisSkladnika, JednostkaMiary) VALUES

(1, 'Mąka Pszenna', 'Mąka pszenna typ 500', 'kg'),

(2, 'Drożdże', 'Drożdże suche', 'g'),

(3, 'Sól', 'Sól kuchenna', 'g'),

(4, 'Cukier', 'Cukier biały', 'g'),

(5, 'Woda', 'Woda mineralna', 'l');

INSERT INTO Magazyn (IdMagazynu, NazwaMagazynu, Adres, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES

(1, 'Magazyn Warszawa', 'Puławska 16', '00-002', 'Warszawa', 7),

(2, 'Magazyn Poznań', 'Podolska 1', '60-002', 'Poznań', 15),

(3, 'Magazyn Kraków', 'Parkowa 32', '30-002', 'Kraków', 6),

(4, 'Magazyn Katowice', 'Szkolna 7', '40-002', 'Katowice', 12),

(5, 'Magazyn Wrocław', 'Zachodnia 7', '50-002', 'Wrocław', 1);

INSERT INTO Dostawcy (IdDostawcy, NazwaDostawcy, Adres, Nip, Telefon, TelefonKomorkowy, Email, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES

(1, 'Młyn Warszawski', 'Grójecka 17', '2345678901', '999000111', '888888881', 'kontakt@mw-warszawski.pl', '00-003', 'Warszawa', 7),

(2, 'Młyn Poznański', 'Młyńska 2', '2345678902', '999000888', '888888882', 'kontakt@mw-poznanski.pl', '60-003', 'Poznań', 15),

(3, 'Młyn Krakowski', 'Klasztorna 114', '2345678903', '999000777', '888888883', 'kontakt@mw-krakowski.pl', '30-003', 'Kraków', 6),

(4, 'Młyn Katowicki', 'Mariacka 90', '2345678904', '999000333', '888888884', 'kontakt@mw-katowicki.pl', '40-003', 'Katowice', 12),

(5, 'Młyn Wrocławski', 'Chopina 2', '2345678905', '999000222', '888888885', 'kontakt@mw-wroclawski.pl', '50-003', 'Wrocław', 1);

-- Wypełnienie tabeli Dostawy

INSERT INTO Dostawy (IdDostawy, IdDostawcy, DataDostawy, KwotaDostawy, IdMagazynu) VALUES

(1, 1, '2024-05-01', 500.00, 1),

(2, 2, '2024-05-02', 600.00, 2),

(3, 3, '2024-05-03', 700.00, 3),

(4, 4, '2024-05-04', 800.00, 4),

(5, 5, '2024-05-05', 900.00, 5);

-- Wypełnienie tabeli SzczegolyDostaw

INSERT INTO SzczegolyDostaw (IdSzczegoluDostawy, IdDostawy, IdSkladnika, Ilosc) VALUES

(1, 1, 1, 100),

(2, 2, 2, 200),

(3, 3, 3, 300),

(4, 4, 4, 400),

(5, 5, 5, 500);

-- Wypełnienie tabeli Samochody

INSERT INTO Samochody (IdSamochodu, NumerRejestracyjny, Vin, RodzajPaliwa, Marka, IdPracownika) VALUES

(1, 'WX12345', '1HGCM82633A123456', 'ON', 'Ford Transit', 1),

(2, 'PO54321', '1HGCM82633A654321', 'PB', 'Renault Master', 2),

(3, 'KR67890', '1HGCM82633A789012', 'ON', 'Mercedes Sprinter', 3),

(4, 'KT09876', '1HGCM82633A987654', 'PB', 'Fiat Ducato', 4),

(5, 'WR56789', '1HGCM82633A123987', 'ON', 'Volkswagen Crafter', 5);

-- Wypełnienie tabeli Zamowienia

INSERT INTO Zamowienia (IdZamowienia, IdSklepu, DataZamowienia, IdSamochodu) VALUES

(1, 1, '2024-05-01', 1),

(2, 2, '2024-05-02', 2),

(3, 3, '2024-05-03', 3),

(4, 4, '2024-05-04', 4),

(5, 5, '2024-05-05', 5);

-- Wypełnienie tabeli SzczegolyZamowien

INSERT INTO SzczegolyZamowien (IdSzczegoluZam, IdZamowienia, IdProduktu, Ilosc) VALUES

(1, 1, 1, 10),

(2, 2, 2, 20),

(3, 3, 3, 30),

(4, 4, 4, 40),

(5, 5, 5, 50);

-- Wypełnienie tabeli Zwroty

INSERT INTO Zwroty (IdZwrotu, IdSklepu, DataZwrotu, Powód, IdPracownika) VALUES

(1, 1, '2024-05-01', 'Niewłaściwy produkt', 1),

(2, 2, '2024-05-02', 'Uszkodzony towar', 2),

(3, 3, '2024-05-03', 'Niezgodność z zamówieniem', 1),

(4, 4, '2024-05-04', 'Zbyt krótki termin ważności', 2),

(5, 5, '2024-05-05', 'Błędna ilość', 3);

-- Wypełnienie tabeli SzczegolyZwrotow

INSERT INTO SzczegolyZwrotow (IdSzczegZwrotu, IdZwrotu, IdProduktu, IdZamowienia, Ilosc) VALUES

(1, 1, 1, 1, 1),

(2, 2, 2, 2, 2),

(3, 3, 3, 3, 3),

(4, 4, 4, 4, 4)

# Wykaz procedur i funkcji do generowania raportów

- up\_RaportZamowienSklepu

- up\_RaportNajczesciejZamawianychProduktow

- uf\_SredniaLiczbaGodzinPracy

- up\_RaportSumaKosztowZamowien

- uf\_LiczbaPracownikowNaStanowisku

- up\_RaportZarobkiPracownikow

# Skrypty tworzące procedury i funkcje do generowania raportów

--Procedury i funkcje do generowania raportów

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_RaportZamowienWszystkichSklepow

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura generująca raport zamówień dla wszystkich sklepów w określonym przedziale czasu

-- Parametry wejściowe:

-- @DataPoczatkowa - data początkowa przedziału czasu

-- @DataKoncowa - data końcowa przedziału czasu

--

-- Parametry wyjściowe: Brak

--

-- Przykład użycia:

-- EXEC up\_RaportZamowienWszystkichSklepow '2024-01-01', '2024-12-31'

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_RaportZamowienWszystkichSklepow

@DataPoczatkowa DATE,

@DataKoncowa DATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @IdSklepu INT;

DECLARE @NazwaSklepu VARCHAR(50);

DECLARE @AdresSklepu VARCHAR(50);

DECLARE @NipSklepu VARCHAR(10);

DECLARE @TelefonSklepu VARCHAR(12);

DECLARE @EmailSklepu VARCHAR(50);

DECLARE @KodPocztowySklepu VARCHAR(6);

DECLARE @MiejscowoscSklepu VARCHAR(35);

DECLARE sklepy\_cursor CURSOR FOR

SELECT IdSklepu, NazwaSklepu, Adres, Nip, Telefon, Email, KodPocztowy, Miejscowosc

FROM Sklepy;

OPEN sklepy\_cursor;

FETCH NEXT FROM sklepy\_cursor INTO @IdSklepu, @NazwaSklepu, @AdresSklepu, @NipSklepu, @TelefonSklepu, @EmailSklepu, @KodPocztowySklepu, @MiejscowoscSklepu;

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT 'Nazwa sklepu: ' + @NazwaSklepu;

PRINT 'Adres: ' + @AdresSklepu;

PRINT 'NIP: ' + @NipSklepu;

PRINT 'Telefon: ' + @TelefonSklepu;

PRINT 'Email: ' + @EmailSklepu;

PRINT 'Kod pocztowy: ' + @KodPocztowySklepu;

PRINT 'Miejscowosc: ' + @MiejscowoscSklepu;

PRINT '--------------------------------------------------';

SELECT IdZamowienia, DataZamowienia

FROM Zamowienia

WHERE IdSklepu = @IdSklepu

AND DataZamowienia BETWEEN @DataPoczatkowa AND @DataKoncowa;

PRINT '';

FETCH NEXT FROM sklepy\_cursor INTO @IdSklepu, @NazwaSklepu, @AdresSklepu, @NipSklepu, @TelefonSklepu, @EmailSklepu, @KodPocztowySklepu, @MiejscowoscSklepu;

END

CLOSE sklepy\_cursor;

DEALLOCATE sklepy\_cursor;

END;

GO

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_RaportNajczesciejZamawianychProduktow

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura generująca raport najczęściej zamawianych produktów

-- Parametry wejściowe: Brak

-- Parametry wyjściowe: Brak

--

-- Przykład użycia: EXEC up\_RaportNajczesciejZamawianychProduktow

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_RaportNajczesciejZamawianychProduktow

AS

BEGIN

SET NOCOUNT ON;

DECLARE @NazwaProduktu NVARCHAR(50);

DECLARE @LiczbaZamowien INT;

DECLARE raport\_cursor CURSOR FOR

SELECT TOP 10

Produkty.NazwaProduktu,

COUNT(SzczegolyZamowien.IdProduktu) AS LiczbaZamowien

FROM SzczegolyZamowien

JOIN Produkty ON SzczegolyZamowien.IdProduktu = Produkty.IdProduktu

GROUP BY SzczegolyZamowien.IdProduktu, Produkty.NazwaProduktu

ORDER BY LiczbaZamowien DESC;

OPEN raport\_cursor;

FETCH NEXT FROM raport\_cursor INTO @NazwaProduktu, @LiczbaZamowien;

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT 'Produkt: ' + @NazwaProduktu + ', Liczba zamówień: ' + CAST(@LiczbaZamowien AS NVARCHAR(10));

FETCH NEXT FROM raport\_cursor INTO @NazwaProduktu, @LiczbaZamowien;

END

CLOSE raport\_cursor;

DEALLOCATE raport\_cursor;

END;

GO

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_GenerujRaportPrzebieguDniaPracy

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura generująca raport przebiegu dnia pracy dla pracowników w określonym przedziale czasu

-- Parametry wejściowe:

-- @DataPoczatkowa - data początkowa przedziału czasu

-- @DataKoncowa - data końcowa przedziału czasu

--

-- Przykład użycia:

-- EXEC up\_GenerujRaportPrzebieguDniaPracy '2024-01-01', '2024-12-24';

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_GenerujRaportPrzebieguDniaPracy

(

@DataPoczatkowa DATE,

@DataKoncowa DATE

)

AS

BEGIN

DECLARE @Imie VARCHAR(50)

DECLARE @Nazwisko VARCHAR(50)

DECLARE @Pesel VARCHAR(11)

DECLARE @Data DATE

DECLARE @LiczbaGodzin INT

DECLARE raport\_cursor CURSOR FOR

SELECT

P.Imie,

P.Nazwisko,

P.Pesel,

PD.Data,

PD.LiczbaGodzin

FROM

Pracownicy P

JOIN

PrzebiegDniaPracy PD ON P.IdPracownika = PD.IdPracownika

WHERE

PD.Data BETWEEN @DataPoczatkowa AND @DataKoncowa

OPEN raport\_cursor

FETCH NEXT FROM raport\_cursor INTO @Imie, @Nazwisko, @Pesel, @Data, @LiczbaGodzin

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT 'Pracownik: ' + @Imie + ' ' + @Nazwisko + ' (PESEL: ' + @Pesel + ')'

PRINT '---------------------------------------------'

PRINT 'Data | Liczba godzin'

PRINT '---------------------------------------------'

PRINT CONVERT(VARCHAR(10), @Data, 120) + ' | ' + CAST(@LiczbaGodzin AS VARCHAR(5))

PRINT ''

FETCH NEXT FROM raport\_cursor INTO @Imie, @Nazwisko, @Pesel, @Data, @LiczbaGodzin

END

CLOSE raport\_cursor

DEALLOCATE raport\_cursor

END

--------------------------------------------------------------------------------------

-- PROCEDURE DEFINITION

-- up\_RaportSumaKosztowZamowien

-- CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura generuje raport sumy kosztów zamówień dla każdego sklepu

--

-- Parametry wejściowe: Brak

-- Parametry wyjściowe: Lista sklepów z sumą kosztów zamówień

--

-- Przykład użycia:

-- EXEC up\_RaportSumaKosztowZamowien;

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_RaportSumaKosztowZamowien

AS

BEGIN

DECLARE @IdSklepu INT

DECLARE @NazwaSklepu VARCHAR(50)

DECLARE @IdProduktu INT

DECLARE @NazwaProduktu VARCHAR(50)

DECLARE @Ilosc INT

DECLARE @Koszt DECIMAL(10, 2)

DECLARE @PrevIdSklepu INT = -1 -- Inicjalizacja zmiennej @PrevIdSklepu

DECLARE raport\_cursor CURSOR FOR

SELECT

Sklepy.IdSklepu,

Sklepy.NazwaSklepu,

SzczegolyZamowien.IdProduktu,

Produkty.NazwaProduktu,

SzczegolyZamowien.Ilosc,

SzczegolyZamowien.Ilosc \* Produkty.CenaProduktu AS Koszt

FROM

SzczegolyZamowien

JOIN

Produkty ON SzczegolyZamowien.IdProduktu = Produkty.IdProduktu

JOIN

Zamowienia ON SzczegolyZamowien.IdZamowienia = Zamowienia.IdZamowienia

JOIN

Sklepy ON Zamowienia.IdSklepu = Sklepy.IdSklepu

ORDER BY

Sklepy.IdSklepu;

OPEN raport\_cursor

FETCH NEXT FROM raport\_cursor INTO @IdSklepu, @NazwaSklepu, @IdProduktu, @NazwaProduktu, @Ilosc, @Koszt

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (@@ROWCOUNT = 1) OR (@IdSklepu <> @PrevIdSklepu)

BEGIN

PRINT '---------------------------------------------'

PRINT ' '

PRINT 'Sklep: ' + @NazwaSklepu

PRINT ' '

PRINT 'Produkty:'

PRINT ' '

PRINT '---------------------------------------------'

END

PRINT ' - ' + @NazwaProduktu + ' (Ilość: ' + CAST(@Ilosc AS VARCHAR(10)) + ', Koszt: ' + CAST(@Koszt AS VARCHAR(20)) + ')'

SET @PrevIdSklepu = @IdSklepu

FETCH NEXT FROM raport\_cursor INTO @IdSklepu, @NazwaSklepu, @IdProduktu, @NazwaProduktu, @Ilosc, @Koszt

END

CLOSE raport\_cursor

DEALLOCATE raport\_cursor

END;

GO

--------------------------------------------------------------------------------------

-- FUNCTION DEFINITION

-- uf\_LiczbaPracownikowNaStanowisku

-- CREATED BY Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Funkcja zwraca tabelę zawierającą liczbę pracowników na każdym stanowisku

--

-- Parametry wejściowe: Brak

-- Parametry wyjściowe: Tabela z liczbą pracowników na każdym stanowisku

--

-- Przykład użycia:

--SELECT \* FROM uf\_LiczbaPracownikowNaStanowisku();

--------------------------------------------------------------------------------------

CREATE FUNCTION uf\_LiczbaPracownikowNaStanowisku()

RETURNS TABLE

AS

RETURN

(

SELECT Stanowiska.IdStanowiska, Stanowiska.Stanowisko, COUNT(\*) AS LiczbaPracownikow

FROM Pracownicy

JOIN Stanowiska ON Pracownicy.IdStanowiska = Stanowiska.IdStanowiska

GROUP BY Stanowiska.IdStanowiska, Stanowiska.Stanowisko

);

GO

--------------------------------------------------------------------------------------

-- PROCEDURE DEFINITION

-- up\_RaportZarobkiPracownikow

-- CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura generuje raport przedstawiający sumę zarobków pracowników za określony okres czasu

--

-- Parametry wejściowe:

-- @DataPoczatkowa - data początkowa okresu

-- @DataKoncowa - data końcowa okresu

--

-- Parametry wyjściowe: Lista pracowników wraz z sumą zarobków

--

-- Przykład użycia:

DECLARE @DataPoczatkowa DATE = '2024-01-01';

DECLARE @DataKoncowa DATE = '2024-12-31';

EXEC up\_RaportZarobkiPracownikow @DataPoczatkowa, @DataKoncowa;

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_RaportZarobkiPracownikow

@DataPoczatkowa DATE,

@DataKoncowa DATE

AS

BEGIN

DECLARE @IdPracownika INT

DECLARE @Imie VARCHAR(50)

DECLARE @Nazwisko VARCHAR(50)

DECLARE @KwotaStawki DECIMAL(10, 2)

DECLARE @LiczbaGodzin INT

DECLARE @SumaZarobkow DECIMAL(10, 2)

DECLARE @PrevIdPracownika INT = -1 -- Inicjalizacja zmiennej @PrevIdPracownika

DECLARE raport\_cursor CURSOR FOR

SELECT

Pracownicy.IdPracownika,

Pracownicy.Imie,

Pracownicy.Nazwisko,

StawkiPracownikow.KwotaStawki,

PrzebiegDniaPracy.LiczbaGodzin

FROM

Pracownicy

JOIN

StawkiPracownikow ON Pracownicy.IdStawki = StawkiPracownikow.IdStawki

JOIN

PrzebiegDniaPracy ON Pracownicy.IdPracownika = PrzebiegDniaPracy.IdPracownika

WHERE

PrzebiegDniaPracy.Data BETWEEN @DataPoczatkowa AND @DataKoncowa

ORDER BY

Pracownicy.IdPracownika;

OPEN raport\_cursor

FETCH NEXT FROM raport\_cursor INTO @IdPracownika, @Imie, @Nazwisko, @KwotaStawki, @LiczbaGodzin

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (@@ROWCOUNT = 1) OR (@IdPracownika <> @PrevIdPracownika)

BEGIN

PRINT '---------------------------------------------'

PRINT 'Pracownik: ' + @Imie + ' ' + @Nazwisko

PRINT 'Suma zarobków: '

END

SET @SumaZarobkow = @KwotaStawki \* @LiczbaGodzin

PRINT ' - ' + CAST(@SumaZarobkow AS VARCHAR(20))

SET @PrevIdPracownika = @IdPracownika

FETCH NEXT FROM raport\_cursor INTO @IdPracownika, @Imie, @Nazwisko, @KwotaStawki, @LiczbaGodzin

END

CLOSE raport\_cursor

DEALLOCATE raport\_cursor

END;

# Wykaz funkcji i procedur do przeglądania danych

- up\_PrzegladajZamowieniaSklepu

- uf\_SredniaCenaProduktowWKategorii

- up\_PrzegladajSzczegolyDostawDlaDostawcy

- up\_PrzegladajPracownikowNaStanowisku

- up\_PrzegladajInformacjePracownika

- up\_PrzegladajNajwiecejPracujacego

- up\_PrzegladajNajmniejPracujacego

# Skrypty tworzące funkcje i procedury do przeglądania danych

--Funkcje i procedury do przeglądania danych

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_PrzegladajZamowieniaSklepu

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura wyświetlająca informacje o wszystkich zamówieniach z danego sklepu

-- Parametry wejściowe:

-- @IdSklepu - identyfikator sklepu

--

-- Parametry wyjściowe: Lista rekordów z zamówieniami dla danego sklepu

--

-- Przykład użycia:

-- DECLARE @IdSklepu INT = 1

-- EXEC up\_PrzegladajZamowieniaSklepu @IdSklepu

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_PrzegladajZamowieniaSklepu

@IdSklepu INT

AS

BEGIN

DECLARE @Zamowienia TABLE (

IdZamowienia INT,

IdSklepu INT,

DataZamowienia DATE,

IdSamochodu INT

);

INSERT INTO @Zamowienia

SELECT IdZamowienia, IdSklepu, DataZamowienia, IdSamochodu

FROM Zamowienia

WHERE IdSklepu = @IdSklepu;

SELECT \* FROM @Zamowienia;

END;

GO

--------------------------------------------------------------------------------------

-- FUNCTION DEFINITION

-- uf\_SredniaCenaProduktowWKategorii

-- CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Funkcja zwracająca średnią cenę produktów w danej kategorii

-- Parametry wejściowe:

-- @IdKategorii - identyfikator kategorii produktów

--

-- Parametry wyjściowe: Średnia cena produktów w danej kategorii

--

-- Przykład użycia:

-- DECLARE @IdKategorii INT = 1

-- SELECT dbo.uf\_SredniaCenaProduktowWKategorii(@IdKategorii) AS SredniaCena

--

--------------------------------------------------------------------------------------

CREATE FUNCTION uf\_SredniaCenaProduktowWKategorii

(

@IdKategorii INT

)

RETURNS DECIMAL(10,2)

AS

BEGIN

DECLARE @SredniaCena DECIMAL(10,2);

SELECT @SredniaCena = AVG(CenaProduktu)

FROM Produkty

WHERE IdKategorii = @IdKategorii;

IF @SredniaCena IS NULL

BEGIN

SET @SredniaCena = 0.00;

END

RETURN @SredniaCena;

END;

GO

--------------------------------------------------------------------------------------

-- PROCEDURE DEFINITION

-- up\_PrzegladajSzczegolyDostawDlaDostawcy

-- CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura wyświetlająca wszystkie szczegóły dostaw dla danego dostawcy

-- Parametry wejściowe:

-- @IdDostawcy - identyfikator dostawcy

--

-- Parametry wyjściowe: Lista szczegółów dostaw dla danego dostawcy

--

-- Przykład użycia:

-- DECLARE @IdDostawcy INT = 1

-- EXEC up\_PrzegladajSzczegolyDostawDlaDostawcy @IdDostawcy

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_PrzegladajSzczegolyDostawDlaDostawcy

@IdDostawcy INT

AS

BEGIN

-- Deklaracja tabeli wynikowej

DECLARE @SzczegolyDostaw TABLE (

IdSzczegoluDostawy INT,

IdDostawy INT,

Produkt VARCHAR(100),

Ilosc INT,

DataDostawy DATE,

KwotaDostawy DECIMAL(10, 2)

);

INSERT INTO @SzczegolyDostaw

SELECT

sd.IdSzczegoluDostawy,

sd.IdDostawy,

sd.Ilosc,

d.DataDostawy,

d.KwotaDostawy

FROM

SzczegolyDostaw sd

INNER JOIN

Dostawy d ON sd.IdDostawy = d.IdDostawy

WHERE

d.IdDostawcy = @IdDostawcy;

SELECT \* FROM @SzczegolyDostaw;

END;

--------------------------------------------------------------------------------------

-- PROCEDURE DEFINITION

-- up\_PrzegladajPracownikowNaStanowisku

-- CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura wyświetlająca pracowników z określonego stanowiska

-- Parametry wejściowe:

-- @Stanowisko - nazwa stanowiska

--

-- Parametry wyjściowe: Lista pracowników na określonym stanowisku

--

-- Przykład użycia:

-- DECLARE @Stanowisko VARCHAR(50) = 'Kierowca'

-- EXEC up\_PrzegladajPracownikowNaStanowisku @Stanowisko

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_PrzegladajPracownikowNaStanowisku

@Stanowisko VARCHAR(50)

AS

BEGIN

DECLARE @PracownicyNaStanowisku TABLE (

IdPracownika INT,

Imie VARCHAR(50),

Nazwisko VARCHAR(50),

IdStanowiska INT

);

INSERT INTO @PracownicyNaStanowisku

SELECT

p.IdPracownika,

p.Imie,

p.Nazwisko,

p.IdStanowiska

FROM

Pracownicy p

INNER JOIN

Stanowiska s ON p.IdStanowiska = s.IdStanowiska

WHERE

s.Stanowisko = @Stanowisko;

SELECT \* FROM @PracownicyNaStanowisku;

END;

GO

--------------------------------------------------------------------------------------

-- FUNCTION DEFINITION

-- uf\_LiczbaProduktowWMagazynieDlaSkladnika

-- CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Funkcja zwracająca liczbę produktów w magazynie dla danego składnika

-- Parametry wejściowe:

-- @IdSkladnika - identyfikator składnika

--

-- Parametry wyjściowe: Liczba produktów w magazynie dla danego składnika

--

-- Przykład użycia:

-- DECLARE @IdSkladnika INT = 1

-- SELECT dbo.uf\_LiczbaProduktowWMagazynieDlaSkladnika(@IdSkladnika) AS LiczbaProduktow

--

--------------------------------------------------------------------------------------

CREATE FUNCTION uf\_LiczbaProduktowWMagazynieDlaSkladnika

(

@IdSkladnika INT

)

RETURNS INT

AS

BEGIN

DECLARE @LiczbaProduktow INT = 0;

SELECT @LiczbaProduktow = SUM(Ilosc)

FROM SzczegolyDostaw

WHERE IdSkladnika = @IdSkladnika;

IF @LiczbaProduktow IS NULL

BEGIN

SET @LiczbaProduktow = 0;

END

RETURN @LiczbaProduktow;

END;

GO

--------------------------------------------------------------------------------------

-- PROCEDURE DEFINITION

-- up\_PrzegladajInformacjePracownika

-- CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura przeglądająca informacje o pracowniku na podstawie numeru PESEL

-- Parametry wejściowe:

-- @Pesel - numer PESEL pracownika

--

-- Parametry wyjściowe: Informacje o pracowniku o podanym numerze PESEL

--

-- Przykład użycia:

-- DECLARE @Pesel VARCHAR(11) = '12345678901'

-- EXEC up\_PrzegladajInformacjePracownika @Pesel

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_PrzegladajInformacjePracownika

@Pesel VARCHAR(11)

AS

BEGIN

-- Deklaracja zmiennej tabeli

DECLARE @InformacjePracownika TABLE (

IdPracownika INT,

Imie VARCHAR(15),

Nazwisko VARCHAR(30),

Pesel VARCHAR(11),

Telefon VARCHAR(12),

TelefonKomorkowy VARCHAR(14),

IdStanowiska INT,

Email VARCHAR(50),

IdStawki INT,

IdPlec INT

);

INSERT INTO @InformacjePracownika

SELECT \* FROM Pracownicy WHERE Pesel = @Pesel;

IF EXISTS (SELECT 1 FROM @InformacjePracownika)

BEGIN

SELECT \* FROM @InformacjePracownika;

END

ELSE

BEGIN

PRINT 'Pracownik o podanym numerze PESEL nie istnieje.';

END

END;

GO

--------------------------------------------------------------------------------------

-- PROCEDURE DEFINITION

-- up\_PrzegladajNajwiecejPracujacego

-- CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura wyświetlająca pracownika, który przepracował najwięcej godzin w określonym przedziale czasowym

-- Parametry wejściowe:

-- @DataPoczatkowa - data początkowa przedziału czasowego

-- @DataKoncowa - data końcowa przedziału czasowego

--

-- Parametry wyjściowe: Informacje o pracowniku, który przepracował najwięcej godzin w określonym przedziale czasowym

--

-- Przykład użycia:

-- DECLARE @DataPoczatkowa DATE = '2024-01-01'

-- DECLARE @DataKoncowa DATE = '2024-01-31'

-- EXEC up\_PrzegladajNajwiecejPracujacego @DataPoczatkowa, @DataKoncowa

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_PrzegladajNajwiecejPracujacego

@DataPoczatkowa DATE,

@DataKoncowa DATE

AS

BEGIN

-- Deklaracja zmiennej tabeli

DECLARE @NajwiecejPracujacy TABLE (

Pracownik VARCHAR(50),

SumaGodzinPracy DECIMAL(10,2)

);

INSERT INTO @NajwiecejPracujacy

SELECT TOP 1 p.Imie + ' ' + p.Nazwisko AS Pracownik,

SUM(pd.LiczbaGodzin) AS SumaGodzinPracy

FROM Pracownicy p

JOIN PrzebiegDniaPracy pd ON p.IdPracownika = pd.IdPracownika

WHERE pd.Data BETWEEN @DataPoczatkowa AND @DataKoncowa

GROUP BY p.Imie, p.Nazwisko

ORDER BY SumaGodzinPracy DESC;

IF EXISTS (SELECT 1 FROM @NajwiecejPracujacy)

BEGIN

SELECT \* FROM @NajwiecejPracujacy;

END

ELSE

BEGIN

PRINT 'Brak danych dla określonego przedziału czasowego.';

END

END;

GO

--------------------------------------------------------------------------------------

-- PROCEDURE DEFINITION

-- up\_PrzegladajNajmniejPracujacego

-- CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura wyświetlająca pracownika, który przepracował najmniej godzin w określonym przedziale czasowym

-- Parametry wejściowe:

-- @DataPoczatkowa - data początkowa przedziału czasowego

-- @DataKoncowa - data końcowa przedziału czasowego

--

-- Parametry wyjściowe: Informacje o pracowniku, który przepracował najmniej godzin w określonym przedziale czasowym

--

-- Przykład użycia:

-- DECLARE @DataPoczatkowa DATE = '2024-01-01'

-- DECLARE @DataKoncowa DATE = '2024-01-31'

-- EXEC up\_PrzegladajNajmniejPracujacego @DataPoczatkowa, @DataKoncowa

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_PrzegladajNajmniejPracujacego

@DataPoczatkowa DATE,

@DataKoncowa DATE

AS

BEGIN

-- Deklaracja zmiennej tabeli

DECLARE @NajmniejPracujacy TABLE (

Pracownik VARCHAR(50),

SumaGodzinPracy DECIMAL(10,2)

);

INSERT INTO @NajmniejPracujacy

SELECT TOP 1 p.Imie + ' ' + p.Nazwisko AS Pracownik,

SUM(pd.LiczbaGodzin) AS SumaGodzinPracy

FROM Pracownicy p

JOIN PrzebiegDniaPracy pd ON p.IdPracownika = pd.IdPracownika

WHERE pd.Data BETWEEN @DataPoczatkowa AND @DataKoncowa

GROUP BY p.Imie, p.Nazwisko

ORDER BY SumaGodzinPracy ASC;

IF EXISTS (SELECT 1 FROM @NajmniejPracujacy)

BEGIN

SELECT \* FROM @NajmniejPracujacy;

END

ELSE

BEGIN

PRINT 'Brak danych dla określonego przedziału czasowego.';

END

END;

GO

# GOWykaz procedur do masowej aktualizacji danych

- up\_ZwiekszCenyProduktowWKategorii

- up\_ZmniejszCenyProduktowWKategorii

- up\_AktualizujPrzebiegDniaPracy

- up\_AktualizujZamowienia

- up\_UstawCeneProduktowNaZero

- up\_DodajStawkeDoStanowiska

# Skrypty tworzące procedury do masowej aktualizacji danych

--Procedury do masowej aktualizacji danych

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_ZwiekszCenyProduktowWKategorii

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura masowej aktualizacji cen produktów w danej kategorii o określony procent

-- Parametry wejściowe:

-- @IdKategorii - identyfikator kategorii produktów, w której mają być zaktualizowane ceny

-- @ProcentWzrostu - procentowy wzrost ceny produktów (np. 10 oznacza wzrost o 10%)

--

-- Parametry wyjściowe: Brak

--

-- Przykład użycia:

-- DECLARE @IdKategorii INT = 1

-- DECLARE @ProcentWzrostu DECIMAL(5,2) = 10.00

-- EXEC up\_AktualizujCenyProduktowWKategorii @IdKategorii, @ProcentWzrostu

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_ZwiekszCenyProduktowWKategorii

@IdKategorii INT,

@ProcentWzrostu DECIMAL(5,2)

AS

BEGIN

-- Aktualizuje ceny produktów w danej kategorii o określony procent

DECLARE @Wzrost DECIMAL(5,2)

SET @Wzrost = 1 + @ProcentWzrostu / 100.0

UPDATE Produkty SET CenaProduktu = CenaProduktu \* @Wzrost WHERE IdKategorii = @IdKategorii;

END;

GO

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_ZmniejszCenyProduktowWKategorii

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura masowej aktualizacji cen produktów w danej kategorii o określony procent

-- Parametry wejściowe:

-- @IdKategorii - identyfikator kategorii produktów, w której mają być zaktualizowane ceny

-- @ProcentZmniejszenia - procentowy spadek ceny produktów (np. 10 oznacza spadek o 10%)

--

-- Parametry wyjściowe: Brak

--

-- Przykład użycia:

-- DECLARE @IdKategorii INT = 1

-- DECLARE @ProcentZmniejszenia DECIMAL(5,2) = 10.00

-- EXEC up\_ZmniejszCenyProduktowWKategorii @IdKategorii, @ProcentZmniejszenia

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_ZmniejszCenyProduktowWKategorii

@IdKategorii INT,

@ProcentZmniejszenia DECIMAL(5,2)

AS

BEGIN

-- Aktualizuje ceny produktów w danej kategorii o określony procent

DECLARE @Spadek DECIMAL(5,2)

SET @Spadek = 1 - @ProcentZmniejszenia / 100.0

UPDATE Produkty SET CenaProduktu = CenaProduktu \* @Spadek WHERE IdKategorii = @IdKategorii;

END;

GO

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_DodajStawkeDoStanowiska

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura dodająca określoną kwotę do stawki dla pracowników na wybranym stanowisku

-- Parametry wejściowe:

-- @IdStanowiska - identyfikator stanowiska, dla którego ma zostać dodana kwota do stawki

-- @KwotaDodatkowa - dodatkowa kwota, która ma zostać dodana do stawki pracowników na tym stanowisku

--

-- Parametry wyjściowe: Brak

--

-- Przykład użycia:

-- DECLARE @IdStanowiska INT = 1

-- DECLARE @KwotaDodatkowa DECIMAL(10,2) = 5.00

-- EXEC up\_DodajStawkeDoStanowiska @IdStanowiska, @KwotaDodatkowa

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_DodajStawkeDoStanowiska

@IdStanowiska INT,

@KwotaDodatkowa DECIMAL(10,2)

AS

BEGIN

-- Dodaje określoną kwotę do stawki pracowników na wybranym stanowisku

UPDATE StawkiPracownikow

SET KwotaStawki = KwotaStawki + @KwotaDodatkowa

WHERE IdStawki IN (SELECT IdStawki FROM Pracownicy WHERE IdStanowiska = @IdStanowiska);

END;

GO

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_AktualizujZamowienia

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura aktualizująca dane o zamówieniach w określonym zakresie czasu

-- Parametry wejściowe:

-- @DataPoczatkowa - data początkowa zakresu czasu (data zamówienia)

-- @DataKoncowa - data końcowa zakresu czasu (data zamówienia)

-- @NowaDataZamowienia - nowa data zamówienia

-- @IloscOpóźnienia - liczba dni opóźnienia

--

-- Parametry wyjściowe: Brak

--

-- Przykład użycia:

-- DECLARE @DataPoczatkowa DATE = '2024-01-01'

-- DECLARE @DataKoncowa DATE = '2024-06-30'

-- DECLARE @NowaDataZamowienia DATE = '2024-07-10'

-- DECLARE @IloscOpóźnienia INT = 5

-- EXEC up\_AktualizujZamowienia @DataPoczatkowa, @DataKoncowa, @NowaDataZamowienia, @NowyStatus, @IloscOpóźnienia

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_AktualizujZamowienia

@DataPoczatkowa DATE,

@DataKoncowa DATE,

@NowaDataZamowienia DATE,

@NowyStatus VARCHAR(50),

@IloscOpóźnienia INT

AS

BEGIN

-- Aktualizuje dane zamówień w określonym zakresie czasu

UPDATE Zamowienia

SET DataZamowienia = DATEADD(day, @IloscOpóźnienia, @NowaDataZamowienia) -- Nowa data zamówienia z uwzględnieniem opóźnienia

WHERE DataZamowienia BETWEEN @DataPoczatkowa AND @DataKoncowa;

END;

GO

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_AktualizujPrzebiegDniaPracy

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura aktualizująca przebieg pracy dla wszystkich pracowników (np. Dodająca liczbe godzin: 0 - dzień wolny)

-- Parametry wejściowe:

-- @Data - data, dla której wprowadzane są zmiany w przebiegu pracy

-- @LiczbaGodzin - liczba godzin przepracowanych w danym dniu

--

-- Parametry wyjściowe: Brak

--

-- Przykład użycia:

-- DECLARE @Data DATE = '2024-07-01'

-- DECLARE @LiczbaGodzin INT = 0

-- EXEC up\_AktualizujPrzebiegDniaPracy @Data, @LiczbaGodzin

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_AktualizujPrzebiegDniaPracy

@Data DATE,

@LiczbaGodzin INT

AS

BEGIN

UPDATE PrzebiegDniaPracy

SET LiczbaGodzin = @LiczbaGodzin

WHERE Data = @Data;

END;

GO

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_AktualizujDaneKontaktoweDostawcy

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura aktualizująca dane kontaktowe dostawcy oraz powiązane dane w zamówieniach

-- Parametry wejściowe:

-- @IdDostawcy - identyfikator dostawcy, którego dane mają być zaktualizowane

-- @NowyTelefon - nowy numer telefonu dostawcy

-- @NowyEmail - nowy adres e-mail dostawcy

--

-- Parametry wyjściowe: Brak

--

-- Przykład użycia:

-- DECLARE @IdDostawcy INT = 1

-- DECLARE @NowyTelefon VARCHAR(12) = '123456789'

-- DECLARE @NowyEmail VARCHAR(50) = 'nowyemail@example.com'

-- EXEC up\_AktualizujDaneKontaktoweDostawcy @IdDostawcy, @NowyTelefon, @NowyEmail

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_AktualizujDaneKontaktoweDostawcy

@IdDostawcy INT,

@NowyTelefon VARCHAR(12),

@NowyEmail VARCHAR(50)

AS

BEGIN

-- Aktualizuje dane kontaktowe dostawcy

UPDATE Dostawcy

SET Telefon = @NowyTelefon,

Email = @NowyEmail

WHERE IdDostawcy = @IdDostawcy;

-- Aktualizuje dane kontaktowe związane z zamówieniami

UPDATE Zamowienia

SET TelefonDostawcy = @NowyTelefon,

EmailDostawcy = @NowyEmail

WHERE IdDostawcy = @IdDostawcy;

END;

GO;

--------------------------------------------------------------------------------------

--PROCEDURE DEFINITION

--up\_ZwiekszStawkiNaStanowisku

--CREATED BY Paweł Wozignój, Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura zwiększająca stawki pracowników na danym stanowisku o określony procent

-- Parametry wejściowe:

-- @IdStanowiska - identyfikator stanowiska, dla którego mają być zwiększone stawki

-- @ProcentWzrostu - procentowy wzrost stawki (np. 10 oznacza wzrost o 10%)

--

-- Parametry wyjściowe: Brak

--

-- Przykład użycia:

-- DECLARE @IdStanowiska INT = 1

-- DECLARE @ProcentWzrostu DECIMAL(5,2) = 10.00

-- EXEC up\_ZwiekszStawkiNaStanowisku @IdStanowiska, @ProcentWzrostu

--

--------------------------------------------------------------------------------------

CREATE PROCEDURE up\_ZwiekszStawkiNaStanowisku

@IdStanowiska INT,

@ProcentWzrostu DECIMAL(5,2)

AS

BEGIN

-- Zwiększa stawki pracowników na danym stanowisku o określony procent

DECLARE @Wzrost DECIMAL(5,2);

SET @Wzrost = 1 + @ProcentWzrostu / 100.0;

UPDATE StawkiPracownikow

SET KwotaStawki = KwotaStawki \* @Wzrost

WHERE IdStawki IN (SELECT IdStawki FROM Pracownicy WHERE IdStanowiska = @IdStanowiska);

END;

GO;

# Skrypty tworzące procedury i triggery do modyfikacji danych

--Procedury i triggery do modyfikacji danych

---PROCEDURE DEFINITION

---up\_DodajPracownika

---CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura dodająca nowego pracownika:

-- Parametry wejściowe:-- @IdPracownika INT - identyfikator pracownika

-- @Imie VARCHAR(15) - imię pracownika

-- @Nazwisko VARCHAR(30) - nazwisko pracownika

-- @Pesel VARCHAR(11) - numer PESEL pracownika

-- @Telefon VARCHAR(12) - numer telefonu pracownika (opcjonalny)

-- @TelefonKomorkowy VARCHAR(14) - numer telefonu komórkowego pracownika (opcjonalny)

-- @IdStanowiska INT - identyfikator stanowiska pracownika

-- @Email VARCHAR(50) - adres email pracownika (opcjonalny)

-- @IdStawki INT - identyfikator stawki pracownika

-- @IdPlec INT - identyfikator płci pracownika

--

-- Parametry wyjściowe:

-- Dodano rekord do tabeli Pracownicy

--

/\*

-- Przykład użycia:

DECLARE @IdPracownika INT = 1

DECLARE @Imie VARCHAR(15) = 'Jan';

DECLARE @Nazwisko VARCHAR(30) = 'Kowalski';

DECLARE @Pesel VARCHAR(11) = '12345678901';

DECLARE @IdStanowiska INT = 1;

DECLARE @IdStawki INT = 1;

DECLARE @IdPlec INT = 1;

EXEC DodajPracownika

@Imie = @Imie,

@Nazwisko = @Nazwisko,

@Pesel = @Pesel,

@IdStanowiska = @IdStanowiska,

@IdStawki = @IdStawki,

@IdPlec = @IdPlec;

\*/

CREATE OR ALTER PROCEDURE up\_DodajPracownika (

@IdPracownika INT,

@Imie VARCHAR(15),

@Nazwisko VARCHAR(30),

@Pesel VARCHAR(11),

@Telefon VARCHAR(12) = NULL,

@TelefonKomorkowy VARCHAR(14) = NULL,

@IdStanowiska INT,

@Email VARCHAR(50) = NULL,

@IdStawki INT,

@IdPlec INT

)

AS

BEGIN TRY

BEGIN

- Sprawdzenie, czy wszystkie pola są uzupełnione

IF @IdPracownika IS NULL OR @Imie IS NULL OR @Nazwisko IS NULL OR @Pesel IS NULL OR @Telefon IS NULL OR @TelefonKomorkowy IS NULL OR @Email IS NULL

BEGIN

RAISERROR('Wszystkie pola muszą być wypełnione.', 16, 1);

RETURN;

END

-- Sprawdzenie długości pól

IF LEN(@Imie) > 15 OR LEN(@Nazwisko) > 30 OR LEN(@Pesel) <> 11 OR LEN(@Telefon) > 12 OR LEN(@TelefonKomorkowy) > 14 OR LEN(@Email) > 50

BEGIN

RAISERROR('Długość pola przekroczyła dopuszczalny limit.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu PESELu

IF NOT @Pesel LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

BEGIN

RAISERROR('Numer PESEL musi składać się z 11 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu telefonu

IF @Telefon IS NOT NULL AND (LEN(@Telefon) <> 9 OR NOT @Telefon LIKE '[0-9]%' OR @Telefon NOT LIKE '\_\_\_\_\_\_\_\_\_')

BEGIN

RAISERROR('Numer telefonu musi składać się z 9 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu telefonu komórkowego

IF @TelefonKomorkowy IS NOT NULL AND (LEN(@TelefonKomorkowy) <> 9 OR NOT @TelefonKomorkowy LIKE '[0-9]%' OR @TelefonKomorkowy NOT LIKE '\_\_\_\_\_\_\_\_\_')

BEGIN

RAISERROR('Numer telefonu komórkowego musi składać się z 9 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu adresu email

IF @Email IS NOT NULL AND NOT @Email LIKE '%@%.%'

BEGIN

RAISERROR('Nieprawidłowy adres email.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy istnieje podane stanowisko

IF NOT EXISTS (SELECT 1 FROM Stanowiska WHERE IdStanowiska = @IdStanowiska)

BEGIN

RAISERROR('Podane stanowisko nie istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy istnieje podana stawka

IF NOT EXISTS (SELECT 1 FROM StawkiPracownikow WHERE IdStawki = @IdStawki)

BEGIN

RAISERROR('Podana stawka nie istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy istnieje podana płeć

IF NOT EXISTS (SELECT 1 FROM Plcie WHERE IdPlec = @IdPlec)

BEGIN

RAISERROR('Podana płeć nie istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy pracownik o podanym PESELu już istnieje

IF EXISTS (SELECT 1 FROM Pracownicy WHERE Pesel = @Pesel)

BEGIN

RAISERROR('Pracownik o podanym PESELu już istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie unikalności adresu e-mail

IF EXISTS (SELECT 1 FROM Pracownicy WHERE Email = @Email)

BEGIN

RAISERROR('Adres e-mail jest już przypisany do innego pracownika.', 16, 1);

RETURN;

END

-- Sprawdzenie unikalności identyfikatora

IF EXISTS (SELECT 1 FROM Pracownicy WHERE IdPracownika = @IdPracownika)

BEGIN

RAISERROR('Pracownik o podanym identyfikatorze już istnieje.', 16,1);

RETURN;

END

-- Dodanie nowego pracownika

INSERT INTO Pracownicy (Imie, Nazwisko, Pesel, Telefon, TelefonKomorkowy, IdStanowiska, Email, IdStawki, IdPlec)

VALUES (@Imie, @Nazwisko, @Pesel, @Telefon, @TelefonKomorkowy, @IdStanowiska, @Email, @IdStawki, @IdPlec);

PRINT 'Nowy pracownik został dodany.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

---PROCEDURE DEFINITION

---up\_UsunPracownika

---CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura usuwająca pracownika:

-- Parametry wejściowe:

-- @IdPracownika INT - identyfikator pracownika do usunięcia

--

-- Parametry wyjściowe:

-- Usunięto rekord z tabeli Pracownicy

--

/\*

-- Przykład użycia:

DECLARE @IdPracownika INT = 1;

EXEC UsunPracownika

@IdPracownika = @IdPracownika;

\*/

CREATE OR ALTER PROCEDURE up\_UsunPracownika (

@IdPracownika INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy pracownik istnieje

IF NOT EXISTS (SELECT 1 FROM Pracownicy WHERE IdPracownika = @IdPracownika)

BEGIN

RAISERROR('Pracownik o podanym identyfikatorze nie istnieje.', 16, 1);

RETURN;

END

-- Usunięcie pracownika

DELETE FROM Pracownicy WHERE IdPracownika = @IdPracownika;

PRINT 'Pracownik został usunięty.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

---PROCEDURE DEFINITION

---up\_AktualizujPracownika

---CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura aktualizująca dane pracownika:

-- Parametry wejściowe:

-- @IdPracownika INT - identyfikator pracownika do aktualizacji

-- @Imie VARCHAR(15) - nowe imię pracownika

-- @Nazwisko VARCHAR(30) - nowe nazwisko pracownika

-- @Pesel VARCHAR(11) - nowy numer PESEL pracownika

-- @Telefon VARCHAR(12) - nowy numer telefonu pracownika

-- @TelefonKomorkowy VARCHAR(14) - nowy numer telefonu komórkowego pracownika

-- @IdStanowiska INT - nowy identyfikator stanowiska pracownika

-- @Email VARCHAR(50) - nowy adres email pracownika

-- @IdStawki INT - nowy identyfikator stawki pracownika

-- @IdPlec INT - nowy identyfikator płci pracownika

--

-- Parametry wyjściowe:

-- Zaktualizowano rekord w tabeli Pracownicy

--

/\*

-- Przykład użycia:

DECLARE @IdPracownika INT = 1;

DECLARE @Imie VARCHAR(15) = 'NoweImie';

DECLARE @Nazwisko VARCHAR(30) = 'NoweNazwisko';

DECLARE @Pesel VARCHAR(11) = '12345678901';

DECLARE @Telefon VARCHAR(12) = '999888777';

DECLARE @TelefonKomorkowy VARCHAR(14) = '666555444';

DECLARE @IdStanowiska INT = 2;

DECLARE @Email VARCHAR(50) = 'nowy@email.pl';

DECLARE @IdStawki INT = 3;

DECLARE @IdPlec INT = 2;

EXEC AktualizujPracownika

@IdPracownika = @IdPracownika,

@Imie = @Imie,

@Nazwisko = @Nazwisko,

@Pesel = @Pesel,

@Telefon = @Telefon,

@TelefonKomorkowy = @TelefonKomorkowy,

@IdStanowiska = @IdStanowiska,

@Email = @Email,

@IdStawki = @IdStawki,

@IdPlec = @IdPlec;

\*/

CREATE OR ALTER PROCEDURE up\_AktualizujPracownika (

@IdPracownika INT,

@Imie VARCHAR(15),

@Nazwisko VARCHAR(30),

@Pesel VARCHAR(11),

@Telefon VARCHAR(12),

@TelefonKomorkowy VARCHAR(14),

@IdStanowiska INT,

@Email VARCHAR(50),

@IdStawki INT,

@IdPlec INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy wszystkie pola są uzupełnione

IF @Imie IS NULL OR @Nazwisko IS NULL OR @Pesel IS NULL OR @Telefon IS NULL OR @TelefonKomorkowy IS NULL OR @Email IS NULL

BEGIN

RAISERROR('Wszystkie pola muszą być wypełnione.', 16, 1);

RETURN;

END

-- Sprawdzenie długości pól

IF LEN(@Imie) > 15 OR LEN(@Nazwisko) > 30 OR LEN(@Pesel) <> 11 OR LEN(@Telefon) > 12 OR LEN(@TelefonKomorkowy) > 14 OR LEN(@Email) > 50

BEGIN

RAISERROR('Długość pola przekroczyła dopuszczalny limit.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu PESELu

IF NOT @Pesel LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

BEGIN

RAISERROR('Numer PESEL musi składać się z 11 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu telefonu

IF @Telefon IS NOT NULL AND (LEN(@Telefon) <> 9 OR NOT @Telefon LIKE '[0-9]%' OR @Telefon NOT LIKE '\_\_\_\_\_\_\_\_\_')

BEGIN

RAISERROR('Numer telefonu musi składać się z 9 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu telefonu komórkowego

IF @TelefonKomorkowy IS NOT NULL AND (LEN(@TelefonKomorkowy) <> 9 OR NOT @TelefonKomorkowy LIKE '[0-9]%' OR @TelefonKomorkowy NOT LIKE '\_\_\_\_\_\_\_\_\_')

BEGIN

RAISERROR('Numer telefonu komórkowego musi składać się z 9 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu adresu email

IF @Email IS NOT NULL AND NOT @Email LIKE '%@%.%'

BEGIN

RAISERROR('Nieprawidłowy adres email.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy istnieje podane stanowisko

IF NOT EXISTS (SELECT 1 FROM Stanowiska WHERE IdStanowiska = @IdStanowiska)

BEGIN

RAISERROR('Podane stanowisko nie istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy istnieje podana stawka

IF NOT EXISTS (SELECT 1 FROM StawkiPracownikow WHERE IdStawki = @IdStawki)

BEGIN

RAISERROR('Podana stawka nie istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy istnieje podana płeć

IF NOT EXISTS (SELECT 1 FROM Plcie WHERE IdPlec = @IdPlec)

BEGIN

RAISERROR('Podana płeć nie istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy pracownik o podanym PESELu już istnieje

IF EXISTS (SELECT 1 FROM Pracownicy WHERE Pesel = @Pesel AND IdPracownika != @IdPracownika)

BEGIN

RAISERROR('Pracownik o podanym PESELu już istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie unikalności adresu e-mail

IF EXISTS (SELECT 1 FROM Pracownicy WHERE Email = @Email AND IdPracownika != @IdPracownika)

BEGIN

RAISERROR('Adres e-mail jest już przypisany do innego pracownika.', 16, 1);

RETURN;

END

IF EXISTS (SELECT 1 FROM Pracownicy WHERE IdPracownika = @IdPracownika AND IdPracownika != @IdPracownika)

BEGIN

RAISERROR('Pracownik o podanym identyfikatorze już istnieje.', 16,1);

RETURN;

END

-- Aktualizacja danych pracownika

UPDATE Pracownicy

SET

Imie = @Imie,

Nazwisko = @Nazwisko,

Pesel = @Pesel,

Telefon = @Telefon,

TelefonKomorkowy = @TelefonKomorkowy,

IdStanowiska = @IdStanowiska,

Email = @Email,

IdStawki = @IdStawki,

IdPlec = @IdPlec

WHERE IdPracownika = @IdPracownika;

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_DodajSklep

-- CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura dodająca nowy sklep:

-- Parametry wejściowe:

-- @IdSklepu INT - identyfikator sklepu

-- @NazwaSklepu VARCHAR(50) - nazwa sklepu

-- @Adres VARCHAR(50) - adres sklepu

-- @Nip VARCHAR(10) - numer NIP sklepu

-- @Telefon VARCHAR(12) - numer telefonu sklepu (opcjonalny)

-- @TelefonKomorkowy VARCHAR(14) - numer telefonu komórkowego sklepu (opcjonalny)

-- @Email VARCHAR(50) - adres email sklepu (opcjonalny)

-- @KodPocztowy VARCHAR(6) - kod pocztowy sklepu

-- @Miejscowosc VARCHAR(35) - miejscowość sklepu

-- @IdWojewodztwa INT - identyfikator województwa sklepu

--

-- Parametry wyjściowe:

-- Dodano rekord do tabeli Sklepy

--

/\*

-- Przykład użycia:

DECLARE @IdSklepu INT = 1;

DECLARE @NazwaSklepu VARCHAR(50) = 'Sklep Spożywczy';

DECLARE @Adres VARCHAR(50) = 'ul. Testowa 1';

DECLARE @Nip VARCHAR(10) = '123-456-78-90';

DECLARE @KodPocztowy VARCHAR(6) = '12-345';

DECLARE @Miejscowosc VARCHAR(35) = 'Kraków';

DECLARE @IdWojewodztwa INT = 1;

EXEC up\_DodajSklep

@IdSklepu = @IdSklepu,

@NazwaSklepu = @NazwaSklepu,

@Adres = @Adres,

@Nip = @Nip,

@KodPocztowy = @KodPocztowy,

@Miejscowosc = @Miejscowosc,

@IdWojewodztwa = @IdWojewodztwa;

\*/

CREATE OR ALTER PROCEDURE up\_DodajSklep (

@IdSklepu INT,

@NazwaSklepu VARCHAR(50),

@Adres VARCHAR(50),

@Nip VARCHAR(10),

@Telefon VARCHAR(12) = NULL,

@TelefonKomorkowy VARCHAR(14) = NULL,

@Email VARCHAR(50) = NULL,

@KodPocztowy VARCHAR(6),

@Miejscowosc VARCHAR(35),

@IdWojewodztwa INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy wszystkie pola są uzupełnione

IF @IdSklepu IS NULL OR @NazwaSklepu IS NULL OR @Adres IS NULL OR @Nip IS NULL OR @KodPocztowy IS NULL OR @Miejscowosc IS NULL OR @IdWojewodztwa IS NULL

BEGIN

RAISERROR('Wszystkie pola muszą być wypełnione.', 16, 1);

RETURN;

END

-- Sprawdzenie długości pól

IF LEN(@NazwaSklepu) > 50 OR LEN(@Adres) > 50 OR LEN(@Nip) <> 10 OR LEN(@Telefon) > 12 OR LEN(@TelefonKomorkowy) > 14 OR LEN(@Email) > 50 OR LEN(@KodPocztowy) <> 6 OR LEN(@Miejscowosc) > 35

BEGIN

RAISERROR('Długość pola przekroczyła dopuszczalny limit.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu NIPu

IF NOT @Nip LIKE '[0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9]-[0-9]'

BEGIN

RAISERROR('Nieprawidłowy format NIPu.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu telefonu

IF @Telefon IS NOT NULL AND (LEN(@Telefon) <> 9 OR NOT @Telefon LIKE '[0-9]%' OR @Telefon NOT LIKE '\_\_\_\_\_\_\_\_\_')

BEGIN

RAISERROR('Numer telefonu musi składać się z 9 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu telefonu komórkowego

IF @TelefonKomorkowy IS NOT NULL AND (LEN(@TelefonKomorkowy) <> 9 OR NOT @TelefonKomorkowy LIKE '[0-9]%' OR @TelefonKomorkowy NOT LIKE '\_\_\_\_\_\_\_\_\_')

BEGIN

RAISERROR('Numer telefonu komórkowego musi składać się z 9 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu adresu email

IF @Email IS NOT NULL AND NOT @Email LIKE '%@%.%'

BEGIN

RAISERROR('Nieprawidłowy adres email.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy podany kod pocztowy jest poprawny

IF NOT @KodPocztowy LIKE '[0-9][0-9]-[0-9][0-9][0-9]'

BEGIN

RAISERROR('Nieprawidłowy format kodu pocztowego.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy istnieje podane województwo

IF NOT EXISTS (SELECT 1 FROM Wojewodztwa WHERE IdWojewodztwa = @IdWojewodztwa)

BEGIN

RAISERROR('Podane województwo nie istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy NIP jest unikalny

IF EXISTS (SELECT 1 FROM Sklepy WHERE Nip = @Nip)

BEGIN

RAISERROR('Sklep o podanym NIPie już istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy nazwa sklepu jest unikalna

IF EXISTS (SELECT 1 FROM Sklepy WHERE NazwaSklepu = @NazwaSklepu)

BEGIN

RAISERROR('Sklep o podanej nazwie już istnieje.', 16, 1);

RETURN;

END

-- Dodanie nowego sklepu

INSERT INTO Sklepy (IdSklepu, NazwaSklepu, Adres, Nip, Telefon, TelefonKomorkowy, Email, KodPocztowy, Miejscowosc, IdWojewodztwa)

VALUES (@IdSklepu, @NazwaSklepu, @Adres, @Nip, @Telefon, @TelefonKomorkowy, @Email, @KodPocztowy, @Miejscowosc, @IdWojewodztwa);

PRINT 'Nowy sklep został dodany.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_UsunSklep

-- CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura usuwająca sklep:

-- Parametry wejściowe:

-- @IdSklepu INT - identyfikator sklepu do usunięcia

--

-- Parametry wyjściowe:

-- Usunięto rekord z tabeli Sklepy

--

/\*

-- Przykład użycia:

DECLARE @IdSklepu INT = 1;

EXEC up\_UsunSklep

@IdSklepu = @IdSklepu;

\*/

CREATE OR ALTER PROCEDURE up\_UsunSklep (

@IdSklepu INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy podany sklep istnieje

IF NOT EXISTS (SELECT 1 FROM Sklepy WHERE IdSklepu = @IdSklepu)

BEGIN

RAISERROR('Sklep o podanym identyfikatorze nie istnieje.', 16, 1);

RETURN;

END

-- Usunięcie sklepu

DELETE FROM Sklepy WHERE IdSklepu = @IdSklepu;

PRINT 'Sklep został usunięty.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_AktualizujSklep

-- CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura aktualizująca dane sklepu:

-- Parametry wejściowe:

-- @IdSklepu INT - identyfikator sklepu do zaktualizowania

-- @NazwaSklepu VARCHAR(50) - nowa nazwa sklepu

-- @Adres VARCHAR(50) - nowy adres sklepu

-- @Nip VARCHAR(10) - nowy numer NIP sklepu

-- @Telefon VARCHAR(12) - nowy numer telefonu sklepu

-- @TelefonKomorkowy VARCHAR(14) - nowy numer telefonu komórkowego sklepu

-- @Email VARCHAR(50) - nowy adres email sklepu

-- @KodPocztowy VARCHAR(6) - nowy kod pocztowy sklepu

-- @Miejscowosc VARCHAR(35) - nowa miejscowość sklepu

-- @IdWojewodztwa INT - nowy identyfikator województwa sklepu

--

-- Parametry wyjściowe:

-- Zaktualizowano rekord w tabeli Sklepy

--

/\*

-- Przykład użycia:

DECLARE @IdSklepu INT = 1;

DECLARE @NazwaSklepu VARCHAR(50) = 'Nowa Nazwa Sklepu';

DECLARE @Adres VARCHAR(50) = 'Nowy Adres 123';

DECLARE @Nip VARCHAR(10) = '987-654-32-10';

DECLARE @Telefon VARCHAR(12) = '123456789';

DECLARE @TelefonKomorkowy VARCHAR(14) = '987654321';

DECLARE @Email VARCHAR(50) = 'nowy@email.com';

DECLARE @KodPocztowy VARCHAR(6) = '11-111';

DECLARE @Miejscowosc VARCHAR(35) = 'Nowe Miasto';

DECLARE @IdWojewodztwa INT = 2;

EXEC up\_AktualizujSklep

@IdSklepu = @IdSklepu,

@NazwaSklepu = @NazwaSklepu,

@Adres = @Adres,

@Nip = @Nip,

@Telefon = @Telefon,

@TelefonKomorkowy = @TelefonKomorkowy,

@Email = @Email,

@KodPocztowy = @KodPocztowy,

@Miejscowosc = @Miejscowosc,

@IdWojewodztwa = @IdWojewodztwa;

\*/

CREATE OR ALTER PROCEDURE up\_AktualizujSklep (

@IdSklepu INT,

@NazwaSklepu VARCHAR(50),

@Adres VARCHAR(50),

@Nip VARCHAR(10),

@Telefon VARCHAR(12),

@TelefonKomorkowy VARCHAR(14),

@Email VARCHAR(50),

@KodPocztowy VARCHAR(6),

@Miejscowosc VARCHAR(35),

@IdWojewodztwa INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy wszystkie pola są uzupełnione

IF @NazwaSklepu IS NULL OR @Adres IS NULL OR @Nip IS NULL OR @KodPocztowy IS NULL OR @Miejscowosc IS NULL OR @IdWojewodztwa IS NULL

BEGIN

RAISERROR('Wszystkie pola muszą być wypełnione.', 16, 1);

RETURN;

END

-- Sprawdzenie długości pól

IF LEN(@NazwaSklepu) > 50 OR LEN(@Adres) > 50 OR LEN(@Nip) > 10 OR LEN(@Telefon) > 12 OR LEN(@TelefonKomorkowy) > 14 OR LEN(@Email) > 50 OR LEN(@KodPocztowy) > 6 OR LEN(@Miejscowosc) > 35

BEGIN

RAISERROR('Długość pola przekroczyła dopuszczalny limit.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu NIPu

IF NOT @Nip LIKE '[0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]'

BEGIN

RAISERROR('Nieprawidłowy numer NIP.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu telefonu

IF @Telefon IS NOT NULL AND (LEN(@Telefon) <> 9 OR NOT @Telefon LIKE '[0-9]%' OR @Telefon NOT LIKE '\_\_\_\_\_\_\_\_\_')

BEGIN

RAISERROR('Numer telefonu musi składać się z 9 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu telefonu komórkowego

IF @TelefonKomorkowy IS NOT NULL AND (LEN(@TelefonKomorkowy) <> 9 OR NOT @TelefonKomorkowy LIKE '[0-9]%' OR @TelefonKomorkowy NOT LIKE '\_\_\_\_\_\_\_\_\_')

BEGIN

RAISERROR('Numer telefonu komórkowego musi składać się z 9 cyfr.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu adresu email

IF @Email IS NOT NULL AND NOT @Email LIKE '%@%.%'

BEGIN

RAISERROR('Nieprawidłowy adres email.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy istnieje podane województwo

IF NOT EXISTS (SELECT 1 FROM Wojewodztwa WHERE IdWojewodztwa = @IdWojewodztwa)

BEGIN

RAISERROR('Podane województwo nie istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie, czy sklep o podanym NIPie już istnieje

IF EXISTS (SELECT 1 FROM Sklepy WHERE Nip = @Nip AND IdSklepu != @IdSklepu)

BEGIN

RAISERROR('Sklep o podanym NIPie już istnieje.', 16, 1);

RETURN;

END

-- Aktualizacja danych sklepu

UPDATE Sklepy

SET

NazwaSklepu = @NazwaSklepu,

Adres = @Adres,

Nip = @Nip,

Telefon = @Telefon,

TelefonKomorkowy = @TelefonKomorkowy,

Email = @Email,

KodPocztowy = @KodPocztowy,

Miejscowosc = @Miejscowosc,

IdWojewodztwa = @IdWojewodztwa

WHERE IdSklepu = @IdSklepu;

PRINT 'Dane sklepu zostały zaktualizowane.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_DodajSamochod

-- CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura dodająca nowy samochód:

-- Parametry wejściowe:

-- @IdSamochodu INT - identyfikator samochodu

-- @NumerRejestracyjny VARCHAR(8) - numer rejestracyjny samochodu

-- @Vin VARCHAR(17) - numer VIN samochodu

-- @RodzajPaliwa CHAR(2) - rodzaj paliwa (np. PB - benzyna, ON - olej napędowy)

-- @Marka VARCHAR(30) - marka samochodu

-- @IdPracownika INT - identyfikator pracownika, który zarządza samochodem

--

-- Parametry wyjściowe:

-- Dodano rekord do tabeli Samochody

--

/\*

-- Przykład użycia:

DECLARE @IdSamochodu INT = 1;

DECLARE @NumerRejestracyjny VARCHAR(8) = 'ABC1234';

DECLARE @Vin VARCHAR(17) = '12345678901234567';

DECLARE @RodzajPaliwa CHAR(2) = 'PB';

DECLARE @Marka VARCHAR(30) = 'Toyota';

DECLARE @IdPracownika INT = 1;

EXEC up\_DodajSamochod

@IdSamochodu = @IdSamochodu,

@NumerRejestracyjny = @NumerRejestracyjny,

@Vin = @Vin,

@RodzajPaliwa = @RodzajPaliwa,

@Marka = @Marka,

@IdPracownika = @IdPracownika;

\*/

CREATE OR ALTER PROCEDURE up\_DodajSamochod (

@IdSamochodu INT,

@NumerRejestracyjny VARCHAR(8),

@Vin VARCHAR(17),

@RodzajPaliwa CHAR(2),

@Marka VARCHAR(30),

@IdPracownika INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy wszystkie pola są uzupełnione

IF @IdSamochodu IS NULL OR @NumerRejestracyjny IS NULL OR @Vin IS NULL OR @RodzajPaliwa IS NULL OR @Marka IS NULL OR @IdPracownika IS NULL

BEGIN

RAISERROR('Wszystkie pola muszą być wypełnione.', 16, 1);

RETURN;

END

-- Sprawdzenie długości pól

IF LEN(@NumerRejestracyjny) <> 8 OR LEN(@Vin) <> 17 OR LEN(@RodzajPaliwa) <> 2 OR LEN(@Marka) > 30

BEGIN

RAISERROR('Długość pola przekroczyła dopuszczalny limit.', 16, 1);

RETURN;

END

-- Sprawdzenie unikalności numeru rejestracyjnego

IF EXISTS (SELECT 1 FROM Samochody WHERE NumerRejestracyjny = @NumerRejestracyjny)

BEGIN

RAISERROR('Samochód o podanym numerze rejestracyjnym już istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie unikalności numeru VIN

IF EXISTS (SELECT 1 FROM Samochody WHERE Vin = @Vin)

BEGIN

RAISERROR('Samochód o podanym numerze VIN już istnieje.', 16, 1);

RETURN;

END

-- Dodanie nowego samochodu

INSERT INTO Samochody (IdSamochodu, NumerRejestracyjny, Vin, RodzajPaliwa, Marka, IdPracownika)

VALUES (@IdSamochodu, @NumerRejestracyjny, @Vin, @RodzajPaliwa, @Marka, @IdPracownika);

PRINT 'Nowy samochód został dodany.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_AktualizujSamochod

-- CREATED BY Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura aktualizująca dane samochodu:

-- Parametry wejściowe:

-- @IdSamochodu INT - identyfikator samochodu do zaktualizowania

-- @NumerRejestracyjny VARCHAR(8) - nowy numer rejestracyjny samochodu

-- @Vin VARCHAR(17) - nowy numer identyfikacyjny pojazdu (VIN)

-- @RodzajPaliwa CHAR(2) - nowy rodzaj paliwa (np. PB - benzyna, ON - olej napędowy)

-- @Marka VARCHAR(30) - nowa marka samochodu

-- @IdPracownika INT - nowy identyfikator pracownika przypisanego do samochodu

--

-- Parametry wyjściowe:

-- Zaktualizowano rekord w tabeli Samochody

--

/\*

-- Przykład użycia:

DECLARE @IdSamochodu INT = 1;

DECLARE @NumerRejestracyjny VARCHAR(8) = 'ABC1234';

DECLARE @Vin VARCHAR(17) = '12345678901234567';

DECLARE @RodzajPaliwa CHAR(2) = 'PB';

DECLARE @Marka VARCHAR(30) = 'Toyota';

DECLARE @IdPracownika INT = 2;

EXEC up\_AktualizujSamochod

@IdSamochodu = @IdSamochodu,

@NumerRejestracyjny = @NumerRejestracyjny,

@Vin = @Vin,

@RodzajPaliwa = @RodzajPaliwa,

@Marka = @Marka,

@IdPracownika = @IdPracownika;

\*/

CREATE OR ALTER PROCEDURE up\_AktualizujSamochod (

@IdSamochodu INT,

@NumerRejestracyjny VARCHAR(8),

@Vin VARCHAR(17),

@RodzajPaliwa CHAR(2),

@Marka VARCHAR(30),

@IdPracownika INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy wszystkie pola są uzupełnione

IF @NumerRejestracyjny IS NULL OR @Vin IS NULL OR @RodzajPaliwa IS NULL OR @Marka IS NULL OR @IdPracownika IS NULL

BEGIN

RAISERROR('Wszystkie pola muszą być wypełnione.', 16, 1);

RETURN;

END

-- Sprawdzenie długości pól

IF LEN(@NumerRejestracyjny) <> 8 OR LEN(@Vin) <> 17 OR LEN(@RodzajPaliwa) <> 2 OR LEN(@Marka) > 30

BEGIN

RAISERROR('Długość pola przekroczyła dopuszczalny limit.', 16, 1);

RETURN;

END

-- Sprawdzenie formatu VIN

IF NOT @Vin LIKE '[0-9A-Z]%'

BEGIN

RAISERROR('Nieprawidłowy numer identyfikacyjny pojazdu (VIN).', 16, 1);

RETURN;

END

-- Sprawdzenie, czy podany pracownik istnieje

IF NOT EXISTS (SELECT 1 FROM Pracownicy WHERE IdPracownika = @IdPracownika)

BEGIN

RAISERROR('Podany pracownik nie istnieje.', 16, 1);

RETURN;

END

-- Sprawdzenie unikalności VIN

IF EXISTS (SELECT 1 FROM Samochody WHERE Vin = @Vin AND IdSamochodu != @IdSamochodu)

BEGIN

RAISERROR('Podany numer identyfikacyjny pojazdu (VIN) już istnieje.', 16, 1);

RETURN;

END

-- Aktualizacja danych samochodu

UPDATE Samochody

SET

NumerRejestracyjny = @NumerRejestracyjny,

Vin = @Vin,

RodzajPaliwa = @RodzajPaliwa,

Marka = @Marka,

IdPracownika = @IdPracownika

WHERE IdSamochodu = @IdSamochodu;

PRINT 'Dane samochodu zostały zaktualizowane.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_UsunSamochod

-- CREATED BY Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura usuwająca samochód:

-- Parametry wejściowe:

-- @IdSamochodu INT - identyfikator samochodu do usunięcia

--

-- Parametry wyjściowe:

-- Usunięto rekord z tabeli Samochody

--

/\*

-- Przykład użycia:

DECLARE @IdSamochodu INT = 1;

EXEC up\_UsunSamochod @IdSamochodu = @IdSamochodu;

\*/

CREATE OR ALTER PROCEDURE up\_UsunSamochod (

@IdSamochodu INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy podany samochód istnieje

IF NOT EXISTS (SELECT 1 FROM Samochody WHERE IdSamochodu = @IdSamochodu)

BEGIN

RAISERROR('Podany samochód nie istnieje.', 16, 1);

RETURN;

END

-- Usunięcie samochodu

DELETE FROM Samochody WHERE IdSamochodu = @IdSamochodu;

PRINT 'Samochód został usunięty.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_DodajZamowienie

-- CREATED BY Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura dodająca nowe zamówienie:

-- Parametry wejściowe:

-- @IdZamowienia INT - identyfikator zamówienia

-- @IdSklepu INT - identyfikator sklepu, z którego pochodzi zamówienie

-- @DataZamowienia DATE - data zamówienia

-- @IdSamochodu INT - identyfikator samochodu, który dowozi zamówienie do sklepu

--

-- Parametry wyjściowe:

-- Dodano rekord do tabeli Zamowienia

--

/\*

-- Przykład użycia:

DECLARE @IdZamowienia INT = 1;

DECLARE @IdSklepu INT = 1;

DECLARE @DataZamowienia DATE = '2024-05-30';

@IdSamochodu INT = 2

EXEC up\_DodajZamowienie

@IdZamowienia = @IdZamowienia,

@IdSklepu = @IdSklepu,

@DataZamowienia = @DataZamowienia;

@IdSamochodu = @IdSamochodu

\*/

CREATE OR ALTER PROCEDURE up\_DodajZamowienie (

@IdZamowienia INT,

@IdSklepu INT,

@DataZamowienia DATE,

@IdSamochodu INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy data zamówienia nie jest przeszła

IF @DataZamowienia < GETDATE()

BEGIN

RAISERROR('Nie można dodać zamówienia z przeszłą datą.', 16, 1);

RETURN;

END

-- Dodanie nowego zamówienia

INSERT INTO Zamowienia (IdZamowienia, IdSklepu, DataZamowienia, IdSamochodu)

VALUES (@IdZamowienia, @IdSklepu, @DataZamowienia, @IdSamochodu);

PRINT 'Nowe zamówienie zostało dodane.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_UsunZamowienie

-- CREATED BY Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura usuwająca zamówienie:

-- Parametry wejściowe:

-- @IdZamowienia INT - identyfikator zamówienia do usunięcia

--

-- Parametry wyjściowe:

-- Usunięto rekord z tabeli Zamowienia

--

/\*

-- Przykład użycia:

DECLARE @IdZamowienia INT = 1;

EXEC up\_UsunZamowienie

@IdZamowienia = @IdZamowienia;

\*/

CREATE OR ALTER PROCEDURE up\_UsunZamowienie (

@IdZamowienia INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy data zamówienia jest przeszła

IF EXISTS (SELECT 1 FROM Zamowienia WHERE IdZamowienia = @IdZamowienia AND DataZamowienia < GETDATE())

BEGIN

RAISERROR('Nie można usunąć zamówienia z przeszłą datą zamówienia.', 16, 1);

RETURN;

END

-- Usunięcie zamówienia

DELETE FROM Zamowienia WHERE IdZamowienia = @IdZamowienia;

PRINT 'Zamówienie zostało usunięte.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_AktualizujZamowienie

-- CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura aktualizująca dane zamówienia:

-- Parametry wejściowe:

-- @IdZamowienia INT - identyfikator zamówienia do zaktualizowania

-- @IdSklepu INT - nowy identyfikator sklepu

-- @DataZamowienia DATE - nowa data zamówienia

-- @IdSamochodu INT - identyfikator samochodu, który dowozi zamówienie do sklepu

--

-- Parametry wyjściowe:

-- Zaktualizowano rekord w tabeli Zamowienia

--

/\*

-- Przykład użycia:

DECLARE @IdZamowienia INT = 1;

DECLARE @IdSklepu INT = 2;

DECLARE @DataZamowienia DATE = '2024-05-31';

DECLARE @IdSamochodu INT = 2

EXEC up\_AktualizujZamowienie

@IdZamowienia = @IdZamowienia,

@IdSklepu = @IdSklepu,

@DataZamowienia = @DataZamowienia,

@IdSamochodu = @IdSamochodu;

\*/

CREATE OR ALTER PROCEDURE up\_AktualizujZamowienie (

@IdZamowienia INT,

@IdSklepu INT,

@DataZamowienia DATE,

@IdSamochodu INT

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy data zamówienia jest przeszła

IF @DataZamowienia < GETDATE()

BEGIN

RAISERROR('Nie można zaktualizować zamówienia na przeszłą datę zamówienia.', 16, 1);

RETURN;

END

-- Aktualizacja danych zamówienia

UPDATE Zamowienia

SET

IdSklepu = @IdSklepu,

DataZamowienia = @DataZamowienia

IdSamochodu = @IdSamochodu

WHERE IdZamowienia = @IdZamowienia;

PRINT 'Dane zamówienia zostały zaktualizowane.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_DodajSzczegolZamowienia

-- CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura dodająca szczegóły zamówienia:

-- Parametry wejściowe:

-- @IdZamowienia INT - identyfikator zamówienia

-- @IdProduktu INT - identyfikator produktu

-- @Ilosc INT - ilość produktów

--

-- Parametry wyjściowe:

-- Dodano rekord do tabeli SzczegolyZamowien

--

/\*

-- Przykład użycia:

DECLARE @IdZamowienia INT = 1;

DECLARE @IdProduktu INT = 2;

DECLARE @Ilosc INT = 10;

EXEC up\_DodajSzczegolZamowienia

@IdZamowienia = @IdZamowienia,

@IdProduktu = @IdProduktu,

@Ilosc = @Ilosc;

\*/

CREATE OR ALTER PROCEDURE up\_DodajSzczegolZamowienia (

@IdZamowienia INT,

@IdProduktu INT,

@Ilosc VARCHAR(10)

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy ilość jest liczbą dodatnią

IF ISNUMERIC(@Ilosc) = 0 OR CAST(@Ilosc AS INT) <= 0

BEGIN

RAISERROR('Ilość musi być dodatnią liczbą całkowitą.', 16, 1);

RETURN;

END

-- Dodanie szczegółów zamówienia

INSERT INTO SzczegolyZamowien (IdZamowienia, IdProduktu, Ilosc)

VALUES (@IdZamowienia, @IdProduktu, @Ilosc);

PRINT 'Szczegóły zamówienia zostały dodane.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_AktualizujSzczegolZamowienia

-- CREATED BY Wojciech Zawieja

--------------------------------------------------------------------------------------

-- Procedura aktualizująca szczegóły zamówienia:

-- Parametry wejściowe:

-- @IdSzczegoluZamowienia INT - identyfikator szczegółu zamówienia do zaktualizowania

-- @Ilosc INT - nowa ilość produktów

--

-- Parametry wyjściowe:

-- Zaktualizowano rekord w tabeli SzczegolyZamowien

--

/\*

-- Przykład użycia:

DECLARE @IdSzczegoluZamowienia INT = 1;

DECLARE @Ilosc INT = 15;

EXEC up\_AktualizujSzczegolZamowienia

@IdSzczegoluZamowienia = @IdSzczegoluZamowienia,

@Ilosc = @Ilosc;

\*/

CREATE OR ALTER PROCEDURE up\_AktualizujSzczegolZamowienia (

@IdSzczegoluZamowienia INT,

@Ilosc VARCHAR(10)

)

AS

BEGIN TRY

BEGIN

-- Sprawdzenie, czy ilość jest liczbą dodatnią

IF ISNUMERIC(@Ilosc) = 0 OR CAST(@Ilosc AS INT) <= 0

BEGIN

RAISERROR('Ilość musi być dodatnią liczbą całkowitą.', 16, 1);

RETURN;

END

-- Aktualizacja szczegółów zamówienia

UPDATE SzczegolyZamowien

SET Ilosc = @Ilosc

WHERE IdSzczegoluZam = @IdSzczegoluZamowienia;

PRINT 'Szczegóły zamówienia zostały zaktualizowane.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- PROCEDURE DEFINITION

-- up\_UsunSzczegolZamowienia

-- CREATED BY Paweł Wozignój

--------------------------------------------------------------------------------------

-- Procedura usuwająca szczegół zamówienia:

-- Parametry wejściowe:

-- @IdSzczegoluZamowienia INT - identyfikator szczegółu zamówienia do usunięcia

--

-- Parametry wyjściowe:

-- Usunięto rekord z tabeli SzczegolyZamowien

--

/\*

-- Przykład użycia:

DECLARE @IdSzczegoluZamowienia INT = 1;

EXEC up\_UsunSzczegolZamowienia

@IdSzczegoluZamowienia = @IdSzczegoluZamowienia;

\*/

CREATE OR ALTER PROCEDURE up\_UsunSzczegolZamowienia (

@IdSzczegoluZamowienia INT

)

AS

BEGIN TRY

BEGIN

-- Usunięcie szczegółu zamówienia

DELETE FROM SzczegolyZamowien

WHERE IdSzczegoluZam = @IdSzczegoluZamowienia;

PRINT 'Szczegół zamówienia został usunięty.';

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

DECLARE @ErrorProcedure NVARCHAR(128);

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE(),

@ErrorProcedure = ERROR\_PROCEDURE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

GO

-- Triggery

/\*

TRIGGER DEFINITION

trg\_SprawdzDuzeZamowienie

CREATED BY Paweł Wozignój

----------------------------------------------------------

Opis:

Trigger sprawdza czy dodawana lub aktualizowana ilość produktu w zamówieniu jest większa niż 100.

Jeśli tak, zapisuje komunikat o dużej ilości produktów w zamówieniu do tabeli Informacje.

Parametry wejściowe:

Insert lub Update w tabeli SzczegolyZamowien

Parametry wyjściowe:

Zapis komunikatu o dużej ilości produktów w zamówieniu do tabeli Informacje

\*/

CREATE OR ALTER TRIGGER tr\_SprawdzDuzeZamowienie

ON SzczegolyZamowien

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @IdZamowienia INT;

DECLARE @Ilosc INT;

SELECT @IdZamowienia = IdZamowienia, @Ilosc = Ilosc

FROM inserted;

IF @Ilosc > 100

BEGIN

DECLARE @Komunikat VARCHAR(500);

SET @Komunikat = 'Uwaga: Zamówienie ' + CAST(@IdZamowienia AS VARCHAR) + ' zawiera dużą ilość produktu!';

INSERT INTO Informacje (Wiadomosc, Gdzie, Data)

VALUES (@Komunikat, 'SzczegolyZamowien', GETDATE());

END

END;

GO

/\*

TRIGGER DEFINITION

trg\_WyswietlNowyZwrot

CREATED BY Paweł Wozignój

----------------------------------------------------------

Opis:

Trigger dodaje komunikat do tabeli Informacje po dodaniu nowego zwrotu produktu, informując o jego szczegółach.

Parametry wejściowe:

Insert w tabeli Zwroty

Parametry wyjściowe:

Komunikat z informacjami o nowym zwrocie produktu

\*/

CREATE OR ALTER TRIGGER tr\_WyswietlNowyZwrot

ON Zwroty

AFTER INSERT

AS

BEGIN

DECLARE @IdZwrotu INT;

DECLARE @DataZwrotu DATE;

DECLARE @Powod VARCHAR(100);

SELECT @IdZwrotu = IdZwrotu, @DataZwrotu = DataZwrotu, @Powod = Powód

FROM inserted;

DECLARE @Komunikat VARCHAR(500);

SET @Komunikat = 'Dodano nowy zwrot produktu: ID zwrotu: ' + CAST(@IdZwrotu AS VARCHAR(10)) +

', Data zwrotu: ' + CONVERT(VARCHAR, @DataZwrotu, 106) +

', Powód: ' + @Powod;

INSERT INTO Informacje (Wiadomosc, Gdzie, Data)

VALUES (@Komunikat, 'Zwroty', GETDATE());

END;

GO

/\*

TRIGGER DEFINITION

trg\_WyswietlDateRealizacjiZamowienia

CREATED BY Wojciech Zawieja

----------------------------------------------------------

Opis:

Trigger wyświetla komunikat po dodaniu lub aktualizacji zamówienia, informując o dacie jego złożenia.

Parametry wejściowe:

Insert lub Update w tabeli Zamowienia

Parametry wyjściowe:

Komunikat: Insert do tabeli Informacje

\*/

CREATE OR ALTER TRIGGER tr\_WyswietlDateRealizacjiZamowienia

ON Zamowienia

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @OrderId INT;

DECLARE @OrderDate DATE;

SELECT TOP 1 @OrderId = IdZamowienia, @OrderDate = DataZamowienia

FROM inserted;

DECLARE @Komunikat VARCHAR(500);

SET @Komunikat = 'Nowe zamówienie o numerze ' + CAST(@OrderId AS VARCHAR(10));

INSERT INTO Informacje (Wiadomosc, Gdzie, Data)

VALUES (@Komunikat, 'Zamowienia', GETDATE());

END;

GO

/\*

TRIGGER DEFINITION

trg\_WyswietlKwoteDostawy

CREATED BY Wojciech Zawieja

----------------------------------------------------------

Opis:

Trigger zapisuje komunikat do tabeli Informacje po dodaniu lub aktualizacji dostawy o kwocie powyżej 1000 złotych.

Dodatkowo zapisuje informację o dacie złożenia dostawy.

Parametry wejściowe:

Insert lub Update w tabeli Dostawy

Parametry wyjściowe:

Komunikat zapisany w tabeli Informacje:

- 'Dostawa o numerze [IdDostawy] wynosi [KwotaDostawy] zł. Uwaga: To duża kwota do zapłaty za dostawę.'

- 'Dostawa o numerze [IdDostawy] została złożona dnia [DataDostawy]'

\*/

CREATE OR ALTER TRIGGER tr\_WyswietlKwoteDostawy

ON Dostawy

AFTER INSERT, UPDATE

AS

BEGIN

-- Wstawienie komunikatu o dużej kwocie dostawy

INSERT INTO Informacje (Wiadomosc, Gdzie, Data)

SELECT

'Dostawa o numerze ' + CAST(IdDostawy AS VARCHAR(10)) + ' wynosi ' + CAST(KwotaDostawy AS VARCHAR(10)) + ' zł. Uwaga: To duża kwota do zapłaty za dostawę.',

'Dostawy',

GETDATE()

FROM inserted

WHERE KwotaDostawy > 1000;

-- Wstawienie informacji o dacie dostawy

INSERT INTO Informacje (Wiadomosc, Gdzie, Data)

SELECT

'Dostawa o numerze ' + CAST(IdDostawy AS VARCHAR(10)) + ' została złożona dnia ' + CONVERT(VARCHAR, DataDostawy, 106),

'Dostawy',

GETDATE()

FROM inserted;

END;

GO

/\*

TRIGGER DEFINITION

trg\_WysokaCenaProduktu

CREATED BY Wojciech Zawieja

----------------------------------------------------------

Opis:

Trigger zapisuje komunikat do tabeli Informacje po dodaniu lub aktualizacji produktu o wysokiej cenie.

Parametry wejściowe:

Insert lub Update w tabeli Produkty

Parametry wyjściowe:

Insert do tabeli Informacje:

'Produkt [NazwaProduktu] (ID: [IdProduktu]) ma wysoką cenę: [CenaProduktu] zł.'

\*/

CREATE OR ALTER TRIGGER tr\_WysokaCenaProduktu

ON Produkty

AFTER INSERT, UPDATE

AS

BEGIN

-- Wstawienie komunikatu o wysokiej cenie produktu

INSERT INTO Informacje (Wiadomosc, Gdzie, Data)

SELECT

'Produkt ' + NazwaProduktu + ' (ID: ' + CAST(IdProduktu AS VARCHAR(10)) + ') ma wysoką cenę: ' + CAST(CenaProduktu AS VARCHAR(10)) + ' zł.',

'Produkty',

GETDATE()

FROM inserted

WHERE CenaProduktu > 50;

END;

GO

# Schemat bazy danych w MS SQL SERVER

Obraz zawierający tekst, diagram, Plan, Rysunek techniczny

Opis wygenerowany automatycznie

# Skrypty SQL tworzące strukturę tabel, powiązania i rejestrujące przykładowe dane(ORACLE)

-- Utworzenie bazy danych

CREATE DATABASE Piekarnia;

-- Użycie bazy danych

ALTER SESSION SET CURRENT\_SCHEMA = Piekarnia;

-- Tworzenie tabel

CREATE TABLE Wojewodztwa (

IdWojewodztwa NUMBER PRIMARY KEY NOT NULL,

Wojewodztwo VARCHAR2(19) NOT NULL

);

CREATE TABLE Sklepy (

IdSklepu NUMBER PRIMARY KEY NOT NULL,

NazwaSklepu VARCHAR2(50) NOT NULL,

Adres VARCHAR2(50) NOT NULL,

Nip VARCHAR2(10) NOT NULL UNIQUE,

Telefon VARCHAR2(12),

TelefonKomorkowy VARCHAR2(14),

Email VARCHAR2(50),

KodPocztowy VARCHAR2(6) NOT NULL,

Miejscowosc VARCHAR2(35) NOT NULL,

IdWojewodztwa NUMBER NOT NULL,

FOREIGN KEY (IdWojewodztwa) REFERENCES Wojewodztwa(IdWojewodztwa)

);

CREATE TABLE KategorieProduktow (

IdKategorii NUMBER PRIMARY KEY NOT NULL,

NazwaKategorii VARCHAR2(30) NOT NULL

);

CREATE TABLE Opakowania (

IdOpakowania NUMBER PRIMARY KEY NOT NULL,

RodzajOpakowania VARCHAR2(30) NOT NULL

);

CREATE TABLE Produkty (

IdProduktu NUMBER PRIMARY KEY NOT NULL,

IdKategorii NUMBER NOT NULL,

IdOpakowania NUMBER NOT NULL,

NazwaProduktu VARCHAR2(30) NOT NULL,

OpisProduktu VARCHAR2(50) NOT NULL,

CenaProduktu NUMBER(10, 2) NOT NULL,

FOREIGN KEY (IdKategorii) REFERENCES KategorieProduktow(IdKategorii),

FOREIGN KEY (IdOpakowania) REFERENCES Opakowania(IdOpakowania)

);

CREATE TABLE Plcie (

IdPlec NUMBER PRIMARY KEY NOT NULL,

Plec VARCHAR2(9) NOT NULL,

SymbolPlci CHAR(1) NOT NULL

);

CREATE TABLE Stanowiska (

IdStanowiska NUMBER PRIMARY KEY NOT NULL,

Stanowisko VARCHAR2(50) NOT NULL

);

CREATE TABLE StawkiPracownikow (

IdStawki NUMBER PRIMARY KEY NOT NULL,

KwotaStawki NUMBER(10, 2) NOT NULL

);

CREATE TABLE Pracownicy (

IdPracownika NUMBER PRIMARY KEY NOT NULL,

Imie VARCHAR2(15) NOT NULL,

Nazwisko VARCHAR2(30) NOT NULL,

Pesel VARCHAR2(11) NOT NULL UNIQUE,

Telefon VARCHAR2(12),

TelefonKomorkowy VARCHAR2(14),

IdStanowiska NUMBER NOT NULL,

Email VARCHAR2(50),

IdStawki NUMBER NOT NULL,

IdPlec NUMBER NOT NULL,

FOREIGN KEY (IdStanowiska) REFERENCES Stanowiska(IdStanowiska),

FOREIGN KEY (IdStawki) REFERENCES StawkiPracownikow(IdStawki),

FOREIGN KEY (IdPlec) REFERENCES Plcie(IdPlec)

);

CREATE TABLE Szkolenia (

IdSzkolenia NUMBER PRIMARY KEY NOT NULL,

NazwaSzkolenia VARCHAR2(50) NOT NULL,

DataSzkolenia DATE NOT NULL

);

CREATE TABLE ZapisaniNaSzkolenia (

IdZapisani NUMBER PRIMARY KEY NOT NULL,

IdPracownika NUMBER NOT NULL,

IdSzkolenia NUMBER NOT NULL,

FOREIGN KEY (IdPracownika) REFERENCES Pracownicy(IdPracownika),

FOREIGN KEY (IdSzkolenia) REFERENCES Szkolenia(IdSzkolenia)

);

CREATE TABLE PrzebiegDniaPracy (

IdPrzebiegu NUMBER PRIMARY KEY NOT NULL,

IdPracownika NUMBER NOT NULL,

Data DATE NOT NULL,

LiczbaGodzin NUMBER NOT NULL,

FOREIGN KEY (IdPracownika) REFERENCES Pracownicy(IdPracownika)

);

CREATE TABLE Zamowienia (

IdZamowienia NUMBER PRIMARY KEY NOT NULL,

IdSklepu NUMBER NOT NULL,

DataZamowienia DATE NOT NULL,

FOREIGN KEY (IdSklepu) REFERENCES Sklepy(IdSklepu)

);

CREATE TABLE SzczegolyZamowien (

IdSzczegoluZam NUMBER PRIMARY KEY NOT NULL,

IdZamowienia NUMBER NOT NULL,

IdProduktu NUMBER NOT NULL,

Ilosc NUMBER NOT NULL,

FOREIGN KEY (IdZamowienia) REFERENCES Zamowienia(IdZamowienia),

FOREIGN KEY (IdProduktu) REFERENCES Produkty(IdProduktu)

);

CREATE TABLE Zwroty (

IdZwrotu NUMBER PRIMARY KEY NOT NULL,

IdSklepu NUMBER NOT NULL,

DataZwrotu DATE NOT NULL,

Powod VARCHAR2(50) NOT NULL,

FOREIGN KEY (IdSklepu) REFERENCES Sklepy(IdSklepu)

);

CREATE TABLE SzczegolyZwrotow (

IdSzczegZwrotu NUMBER PRIMARY KEY NOT NULL,

IdZwrotu NUMBER NOT NULL,

IdProduktu NUMBER NOT NULL,

IdZamowienia NUMBER NOT NULL,

Ilosc NUMBER NOT NULL,

FOREIGN KEY (IdZwrotu) REFERENCES Zwroty(IdZwrotu),

FOREIGN KEY (IdProduktu) REFERENCES Produkty(IdProduktu),

FOREIGN KEY (IdZamowienia) REFERENCES Zamowienia(IdZamowienia)

);

CREATE TABLE Magazyn (

IdMagazynu NUMBER PRIMARY KEY NOT NULL,

NazwaMagazynu VARCHAR2(50) NOT NULL,

Adres VARCHAR2(50) NOT NULL,

KodPocztowy VARCHAR2(6) NOT NULL,

Miejscowosc VARCHAR2(35) NOT NULL,

IdWojewodztwa NUMBER NOT NULL,

FOREIGN KEY (IdWojewodztwa) REFERENCES Wojewodztwa(IdWojewodztwa)

);

CREATE TABLE Skladniki (

IdSkladnika NUMBER PRIMARY KEY NOT NULL,

NazwaSkladnika VARCHAR2(30) NOT NULL,

OpisSkladnika VARCHAR2(50),

JednostkaMiary VARCHAR2(5) NOT NULL

);

CREATE TABLE Dostawcy (

IdDostawcy NUMBER PRIMARY KEY,

NazwaDostawcy VARCHAR2(50) NOT NULL,

Adres VARCHAR2(50) NOT NULL,

Nip VARCHAR2(10) NOT NULL UNIQUE,

Telefon VARCHAR2(12),

TelefonKomorkowy VARCHAR2(14),

Email VARCHAR2(50),

KodPocztowy VARCHAR2(6) NOT NULL,

Miejscowosc VARCHAR2(35) NOT NULL,

IdWojewodztwa NUMBER NOT NULL,

FOREIGN KEY (IdWojewodztwa) REFERENCES Wojewodztwa(IdWojewodztwa)

);

CREATE TABLE Dostawy (

IdDostawy NUMBER PRIMARY KEY,

IdDostawcy NUMBER NOT NULL,

DataDostawy DATE NOT NULL,

KwotaDostawy NUMBER(10, 2) NOT NULL,

IdMagazynu NUMBER NOT NULL,

FOREIGN KEY (IdDostawcy) REFERENCES Dostawcy(IdDostawcy),

FOREIGN KEY (IdMagazynu) REFERENCES Magazyn(IdMagazynu)

);

CREATE TABLE SzczegolyDostaw (

IdSzczegoluDostawy NUMBER PRIMARY KEY,

IdDostawy NUMBER NOT NULL,

IdSkladnika NUMBER NOT NULL,

Ilosc NUMBER NOT NULL,

FOREIGN KEY (IdDostawy) REFERENCES Dostawy(IdDostawy),

FOREIGN KEY (IdSkladnika) REFERENCES Skladniki(IdSkładnika)

);

CREATE TABLE Samochody (

IdSamochodu NUMBER PRIMARY KEY,

NumerRejestracyjny VARCHAR2(8) NOT NULL UNIQUE,

Vin VARCHAR2(17) NOT NULL UNIQUE,

RodzajPaliwa CHAR(2) NOT NULL,

Marka VARCHAR2(30) NOT NULL,

IdPracownika NUMBER NOT NULL,

FOREIGN KEY (IdPracownika) REFERENCES Pracownicy(IdPracownika)

);

INSERT INTO Wojewodztwa (IdWojewodztwa, Wojewodztwo) VALUES

(1, 'dolnośląskie'),

(2, 'kujawsko-pomorskie'),

(3, 'lubelskie'),

(4, 'lubuskie'),

(5, 'łódzkie'),

(6, 'małopolskie'),

(7, 'mazowieckie'),

(8, 'opolskie'),

(9, 'podkarpackie'),

(10, 'podlaskie'),

(11, 'pomorskie'),

(12, 'śląskie'),

(13, 'świętokrzyskie'),

(14, 'warmińsko-mazurskie'),

(15, 'wielkopolskie'),

(16, 'zachodniopomorskie');

INSERT INTO Sklepy (IdSklepu, NazwaSklepu, Adres, Nip, Telefon, TelefonKomorkowy, Email, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES

(1, 'Piekarnia Warszawska', 'Grójecka 7', '1234567890', '111222333', '833199222', 'warszawska@piekarnia.pl', '00-001', 'Warszawa', 6),

(2, 'Piekarnia Mąka', 'Kopcińskiego 64', '1234567891', '123123123', '033823182', 'maka@piekarnia.pl', '60-001', 'Łódź', 5),

(3, 'Piekarnia Pawełek', 'Legionów 66', '1234567892', '456456456', '666777111', 'pawelek@piekarnia.pl', '30-001', 'Wołomin', 6),

(4, 'Piekarnia Putka', 'Piotrkowska 193', '1234567893', '444555666', '166166611', 'putka@piekarnia.pl', '40-001', 'Łódź', 5),

(5, 'Piekarnia Kłos', 'Żelazna 1', '1234567894', '787787787', '957217222', 'klos@piekarnia.pl', '50-001', 'Katowica', 12);

INSERT INTO KategorieProduktow (IdKategorii, NazwaKategorii) VALUES

(1, 'Chleby'),

(2, 'Bułki'),

(3, 'Ciasta'),

(4, 'Przekąski Słone'),

(5, 'Przekąski Słodkie');

INSERT INTO Opakowania (IdOpakowania, RodzajOpakowania) VALUES

(1, 'Papier'),

(2, 'Folia'),

(3, 'Pudełko kartonowe'),

(4, 'Torebka papierowa'),

(5, 'Plastik');

INSERT INTO Produkty (IdProduktu, IdKategorii, IdOpakowania, NazwaProduktu, OpisProduktu, CenaProduktu) VALUES

(1, 1, 1, 'Chleb Żytni', 'Chleb żytni na zakwasie', 4.50),

(2, 2, 4, 'Bułka Kajzerka', 'Tradycyjna bułka kajzerka', 0.60),

(3, 3, 3, 'Ciasto Czekoladowe', 'Ciasto czekoladowe z polewą', 12.00),

(4, 4, 2, 'Bagietka Czosnkowa', 'Bagietka czosnkowa', 3.00),

(5, 5, 2, 'Muffin Czekoladowy', 'Muffin czekoladowy z kawałkami czekolady', 2.50);

INSERT INTO Plcie (IdPlec, Plec, SymbolPlci) VALUES

(1, 'Mężczyzna', 'M'),

(2, 'Kobieta', 'K');

INSERT INTO Stanowiska (IdStanowiska, Stanowisko) VALUES

(1, 'Sprzedawca'),

(2, 'Piekarz'),

(3, 'Kierownik Sklepu'),

(4, 'Magazynier'),

(5, 'Kierowca');

INSERT INTO StawkiPracownikow (IdStawki, KwotaStawki) VALUES

(1, 15.00),

(2, 18.00),

(3, 20.00),

(4, 22.00),

(5, 25.00);

INSERT INTO Szkolenia (IdSzkolenia, NazwaSzkolenia, DataSzkolenia) VALUES

(1, 'Bezpieczeństwo pracy', TO\_DATE('2024-06-01', 'YYYY-MM-DD')),

(2, 'Obsługa klienta', TO\_DATE('2024-06-02', 'YYYY-MM-DD')),

(3, 'Techniki sprzedaży', TO\_DATE('2024-06-03', 'YYYY-MM-DD')),

(4, 'Przygotowanie pieczywa', TO\_DATE('2024-06-04', 'YYYY-MM-DD')),

(5, 'Zarządzanie zespołem', TO\_DATE('2024-06-05', 'YYYY-MM-DD'));

INSERT INTO Pracownicy (IdPracownika, Imie, Nazwisko, Pesel, Telefon, TelefonKomorkowy, IdStanowiska, Email, IdStawki, IdPlec) VALUES

(1, 'Jan', 'Kowalski', '12345678901', '222333444', '666777888', 1, 'jan.kowalski@piekarnia.pl', 1, 1),

(2, 'Anna', 'Nowak', '12345678902', '222333445', '666777889', 2, 'anna.nowak@piekarnia.pl', 2, 2),

(3, 'Piotr', 'Wiśniewski', '12345678903', '222333446', '666777890', 3, 'piotr.wisniewski@piekarnia.pl', 3, 1),

(4, 'Katarzyna', 'Wójcik', '12345678904', '222333447', '666777891', 4, 'katarzyna.wojcik@piekarnia.pl', 4, 2),

(5, 'Tomasz', 'Kowalczyk', '12345678905', '222333448', '666777892', 5, 'tomasz.kowalczyk@piekarnia.pl', 5, 1);

INSERT INTO ZapisaniNaSzkolenia (IdZapisani, IdPracownika, IdSzkolenia) VALUES

(1, 1, 1),

(2, 2, 2),

(3, 3, 3),

(4, 4, 4),

(5, 5, 5);

INSERT INTO PrzebiegDniaPracy (IdPrzebiegu, IdPracownika, Data, LiczbaGodzin) VALUES

(1, 1, TO\_DATE('2024-05-01', 'YYYY-MM-DD'), 8),

(2, 2, TO\_DATE('2024-05-01', 'YYYY-MM-DD'), 8),

(3, 3, TO\_DATE('2024-05-01', 'YYYY-MM-DD'), 8),

(4, 4, TO\_DATE('2024-05-01', 'YYYY-MM-DD'), 8),

(5, 5, TO\_DATE('2024-05-01', 'YYYY-MM-DD'), 8);

INSERT INTO Skladniki (IdSkladnika, NazwaSkladnika, OpisSkladnika, JednostkaMiary) VALUES

(1, 'Mąka Pszenna', 'Mąka pszenna typ 500', 'kg'),

(2, 'Drożdże', 'Drożdże suche', 'g'),

(3, 'Sól', 'Sól kuchenna', 'g'),

(4, 'Cukier', 'Cukier biały', 'g'),

(5, 'Woda', 'Woda mineralna', 'l');

-- Insert data into Magazyn

INSERT ALL

INTO Magazyn (IdMagazynu, NazwaMagazynu, Adres, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (1, 'Magazyn Warszawa', 'Puławska 16', '00-002', 'Warszawa', 7)

INTO Magazyn (IdMagazynu, NazwaMagazynu, Adres, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (2, 'Magazyn Poznań', 'Podolska 1', '60-002', 'Poznań', 15)

INTO Magazyn (IdMagazynu, NazwaMagazynu, Adres, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (3, 'Magazyn Kraków', 'Parkowa 32', '30-002', 'Kraków', 6)

INTO Magazyn (IdMagazynu, NazwaMagazynu, Adres, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (4, 'Magazyn Katowice', 'Szkolna 7', '40-002', 'Katowice', 12)

INTO Magazyn (IdMagazynu, NazwaMagazynu, Adres, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (5, 'Magazyn Wrocław', 'Zachodnia 7', '50-002', 'Wrocław', 1)

SELECT \* FROM dual;

-- Insert data into Dostawcy

INSERT ALL

INTO Dostawcy (IdDostawcy, NazwaDostawcy, Adres, Nip, Telefon, TelefonKomorkowy, Email, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (1, 'Młyn Warszawski', 'Grójecka 17', '2345678901', '999000111', '888888881', 'kontakt@mw-warszawski.pl', '00-003', 'Warszawa', 7)

INTO Dostawcy (IdDostawcy, NazwaDostawcy, Adres, Nip, Telefon, TelefonKomorkowy, Email, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (2, 'Młyn Poznański', 'Młyńska 2', '2345678902', '999000888', '888888882', 'kontakt@mw-poznanski.pl', '60-003', 'Poznań', 15)

INTO Dostawcy (IdDostawcy, NazwaDostawcy, Adres, Nip, Telefon, TelefonKomorkowy, Email, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (3, 'Młyn Krakowski', 'Klasztorna 114', '2345678903', '999000777', '888888883', 'kontakt@mw-krakowski.pl', '30-003', 'Kraków', 6)

INTO Dostawcy (IdDostawcy, NazwaDostawcy, Adres, Nip, Telefon, TelefonKomorkowy, Email, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (4, 'Młyn Katowicki', 'Mariacka 90', '2345678904', '999000333', '888888884', 'kontakt@mw-katowicki.pl', '40-003', 'Katowice', 12)

INTO Dostawcy (IdDostawcy, NazwaDostawcy, Adres, Nip, Telefon, TelefonKomorkowy, Email, KodPocztowy, Miejscowosc, IdWojewodztwa) VALUES (5, 'Młyn Wrocławski', 'Chopina 2', '2345678905', '999000222', '888888885', 'kontakt@mw-wroclawski.pl', '50-003', 'Wrocław', 1)

SELECT \* FROM dual;

-- Insert data into Dostawy

INSERT ALL

INTO Dostawy (IdDostawy, IdDostawcy, DataDostawy, KwotaDostawy, IdMagazynu) VALUES (1, 1, DATE '2024-05-01', 500.00, 1)

INTO Dostawy (IdDostawy, IdDostawcy, DataDostawy, KwotaDostawy, IdMagazynu) VALUES (2, 2, DATE '2024-05-02', 600.00, 2)

INTO Dostawy (IdDostawy, IdDostawcy, DataDostawy, KwotaDostawy, IdMagazynu) VALUES (3, 3, DATE '2024-05-03', 700.00, 3)

INTO Dostawy (IdDostawy, IdDostawcy, DataDostawy, KwotaDostawy, IdMagazynu) VALUES (4, 4, DATE '2024-05-04', 800.00, 4)

INTO Dostawy (IdDostawy, IdDostawcy, DataDostawy, KwotaDostawy, IdMagaz

-- Insert data into SzczegolyDostaw

INSERT ALL

INTO SzczegolyDostaw (IdSzczegoluDostawy, IdDostawy, IdSkladnika, Ilosc) VALUES (1, 1, 1, 100)

INTO SzczegolyDostaw (IdSzczegoluDostawy, IdDostawy, IdSkladnika, Ilosc) VALUES (2, 2, 2, 200)

INTO SzczegolyDostaw (IdSzczegoluDostawy, IdDostawy, IdSkladnika, Ilosc) VALUES (3, 3, 3, 300)

INTO SzczegolyDostaw (IdSzczegoluDostawy, IdDostawy, IdSkladnika, Ilosc) VALUES (4, 4, 4, 400)

INTO SzczegolyDostaw (IdSzczegoluDostawy, IdDostawy, IdSkladnika, Ilosc) VALUES (5, 5, 5, 500)

SELECT \* FROM dual;

-- Insert data into Zamowienia

INSERT ALL

INTO Zamowienia (IdZamowienia, IdSklepu, DataZamowienia) VALUES (1, 1, DATE '2024-05-01')

INTO Zamowienia (IdZamowienia, IdSklepu, DataZamowienia) VALUES (2, 2, DATE '2024-05-02')

INTO Zamowienia (IdZamowienia, IdSklepu, DataZamowienia) VALUES (3, 3, DATE '2024-05-03')

INTO Zamowienia (IdZamowienia, IdSklepu, DataZamowienia) VALUES (4, 4, DATE '2024-05-04')

INTO Zamowienia (IdZamowienia, IdSklepu, DataZamowienia) VALUES (5, 5, DATE '2024-05-05')

SELECT \* FROM dual;

-- Insert data into SzczegolyZamowien

INSERT ALL

INTO SzczegolyZamowien (IdSzczegoluZam, IdZamowienia, IdProduktu, Ilosc) VALUES (1, 1, 1, 10)

INTO SzczegolyZamowien (IdSzczegoluZam, IdZamowienia, IdProduktu, Ilosc) VALUES (2, 2, 2, 20)

INTO SzczegolyZamowien (IdSzczegoluZam, IdZamowienia, IdProduktu, Ilosc) VALUES (3, 3, 3, 30)

INTO SzczegolyZamowien (IdSzczegoluZam, IdZamowienia, IdProduktu, Ilosc) VALUES (4, 4, 4, 40)

INTO SzczegolyZamowien (IdSzczegoluZam, IdZamowienia, IdProduktu, Ilosc) VALUES (5, 5, 5, 50)

SELECT \* FROM dual;

-- Insert data into Zwroty

INSERT ALL

INTO Zwroty (IdZwrotu, IdSklepu, DataZwrotu, Powód) VALUES (1, 1, DATE '2024-05-01', 'Niewłaściwy produkt')

INTO Zwroty (IdZwrotu, IdSklepu, DataZwrotu, Powód) VALUES (2, 2, DATE '2024-05-02', 'Uszkodzony towar')

INTO Zwroty (IdZwrotu, IdSklepu, DataZwrotu, Powód) VALUES (3, 3, DATE '2024-05-03', 'Niezgodność z zamówieniem')

INTO Zwroty (IdZwrotu, IdSklepu, DataZwrotu, Powód) VALUES (4, 4, DATE '2024-05-04', 'Zbyt krótki termin ważności')

INTO Zwroty (IdZwrotu, IdSklepu, DataZwrotu, Powód) VALUES (5, 5, DATE '2024-05-05', 'Błędna ilość')

SELECT \* FROM dual;

INSERT ALL

INTO SzczegolyZwrotow (IdSzczegZwrotu, IdZwrotu, IdProduktu, IdZamowienia, Ilosc) VALUES (1, 1, 1, 1, 1)

INTO SzczegolyZwrotow (IdSzczegZwrotu, IdZwrotu, IdProduktu, IdZamowienia, Ilosc) VALUES (2, 2, 2, 2, 2)

INTO SzczegolyZwrotow (IdSzczegZwrotu, IdZwrotu, IdProduktu, IdZamowienia, Ilosc) VALUES (3, 3, 3, 3, 3)

INTO SzczegolyZwrotow (IdSzczegZwrotu, IdZwrotu, IdProduktu, IdZamowienia, Ilosc) VALUES (4, 4, 4, 4, 4)

INTO SzczegolyZwrotow (IdSzczegZwrotu, IdZwrotu, IdProduktu, IdZamowienia, Ilosc) VALUES (5, 5, 5, 5, 5)

SELECT \* FROM dual;

-- Insert data into Samochody

INSERT ALL

INTO Samochody (IdSamochodu, NumerRejestracyjny, Vin, RodzajPaliwa, Marka, IdPracownika) VALUES (1, 'WX12345', '1HGCM82633A123456', 'ON', 'Ford Transit', 1)

INTO Samochody (IdSamochodu, NumerRejestracyjny, Vin, RodzajPaliwa, Marka, IdPracownika) VALUES (2, 'PO54321', '1HGCM82633A654321', 'PB', 'Renault Master', 2)

INTO Samochody (IdSamochodu, NumerRejestracyjny, Vin, RodzajPaliwa, Marka, IdPracownika) VALUES (3, 'KR67890', '1HGCM82633A789012', 'ON', 'Mercedes Sprinter', 3)

INTO Samochody (IdSamochodu, NumerRejestracyjny, Vin, RodzajPaliwa, Marka, IdPracownika) VALUES (4, 'KT09876', '1HGCM82633A987654', 'PB', 'Fiat Ducato', 4)

INTO Samochody (IdSamochodu, NumerRejestracyjny, Vin, RodzajPaliwa, Marka, IdPracownika) VALUES (5, 'WR56789', '1HGCM82633A123987', 'ON', 'Volkswagen Crafter', 5)

SELECT \* FROM dual;

# Wnioski

* Implementacja bazy danych w piekarni jest kluczowa dla usprawnienia zarządzania informacjami o procesów związanych z prowadzeniem firmy piekarniczej.
* Baza danych umożliwia efektywne gromadzenie, organizację i aktualizację danych dotyczących różnych struktur piekarni, w tym informacji o pracownikach, dostawach, zamówieniach, składnikach, itp.
* Dzięki bazie danych personel piekarni będzie mógł szybko i łatwo wyszukiwać np. informacje o szczegółach zamówień (jaki produkt oraz jaka ilość), co usprawni proces wykonywanych czynności i przyśpieszy w tym przypadku realizacje zamówień.
* Baza danych ułatwi również monitorowanie stanu zwrotów niesprzedanych, bądź produktów z wadami, zapisywanie na szkolenia, rejestracje dnia roboczego pracownika ( ilość godzin przepracowanych w danym dniu), co ułatwi wyliczanie wynagrodzeń oraz organizacje prac

# Podsumowanie

Wprowadzenie bazy danych do piekarni stanowi znaczący krok w usprawnianiu procesów zarządzania informacjami oraz podniesieniu jakości usług realizowanych przez piekarnie. Baza danych umożliwia skuteczne gromadzenie, organizację i aktualizację danych dotyczących sklepów, do których są rozwożone produkty z piekarni, co ma kluczowe znaczenie dla skutecznego zarządzania piekarnią.

Dzięki bazie danych personel piekarni będzie mógł szybko i łatwo wyszukiwać np. informacje o szczegółach zamówień (jaki produkt oraz jaka ilość), co usprawni proces wykonywanych czynności i przyśpieszy w tym przypadku realizacje zamówień. Baza danych ułatwi również monitorowanie stanu zwrotów niesprzedanych, bądź produktów z wadami, zapisywanie na szkolenia, rejestracje dnia roboczego pracownika (ilość godzin przepracowanych w danym dniu), co ułatwi wyliczanie wynagrodzeń oraz organizację pracy. Dzięki temu systemowi piekarnia może efektywniej zarządzać swoimi zasobami oraz poprawić jakość obsługi klienta.

Oprócz tego, baza danych umożliwia także analizę danych, co pozwala na identyfikację trendów, preferencji klientów oraz optymalizację procesów produkcyjnych i sprzedażowych. Możliwe jest również generowanie raportów i statystyk, które wspierają podejmowanie decyzji biznesowych oraz planowanie strategii rozwoju firmy. W ten sposób, baza danych staje się niezbędnym narzędziem dla piekarni, umożliwiającym skuteczne zarządzanie i rozwój przedsiębiorstwa.Początek formularza