**Tytuł projektu:** Baza danych firmy sprzedającej nieruchomości

**Opis projektu:** Projekt zakłada stworzenie kompleksowej bazy danych dla firmy zajmującej się sprzedażą i wynajmem nieruchomości. Baza ta będzie przechowywać informacje o nieruchomościach, klientach, agentach, deweloperach oraz transakcjach między nimi. Celem jest umożliwienie efektywnego zarządzania ofertami nieruchomości, klientami, monitorowanie aktywności agentów i realizowanych transakcji, a także śledzenie umów najmu i prowizji wypłacanych agentom. Dodatkowo, baza umożliwia rejestrowanie szczegółowych informacji o deweloperach współpracujących z firmą.

**Zdania SQL, które utworzą bazę danych, z krótkim opisem słownym każdej tabeli:**

1. **Miasta**: Tabela przechowująca nazwy miast, w których znajdują się nieruchomości, wraz z krajem i liczbą mieszkańców.
2. **TypyNieruchomosci**: Tabela definiująca różne typy nieruchomości, takie jak mieszkania, domy, działki, biura, lokale użytkowe, z dodatkowym opisem i statusem aktywności.
3. **StatusyTransakcji**: Tabela określająca różne statusy transakcji nieruchomościami, np. "W trakcie", "Zakończona", "Anulowana", z opisem i datą dodania.
4. **Nieruchomosci**: Tabela zawierająca szczegółowe informacje o nieruchomościach, takie jak adres, cena, powierzchnia, data dodania do bazy oraz status.
5. **Klienci**: Tabela przechowująca dane o klientach, w tym imię, nazwisko, email, telefon oraz miasto zamieszkania.
6. **Agenci**: Tabela rejestrująca informacje o agentach sprzedających nieruchomości, w tym imię, nazwisko, email, telefon oraz miasto pracy.
7. **Transakcje**: Tabela śledząca informacje o każdej transakcji, takie jak nieruchomość, klient, agent, data transakcji oraz kwota transakcji.
8. **ProwizjeAgentow**: Tabela przechowująca informacje o prowizjach wypłacanych agentom za przeprowadzone transakcje.
9. **Deweloperzy**: Tabela zawierająca informacje o deweloperach, w tym nazwa, adres, email, telefon oraz miasto, w którym prowadzą działalność.
10. **UmowyNajmu**: Tabela przechowująca informacje o umowach najmu, takie jak ID nieruchomości, ID klienta, data rozpoczęcia i zakończenia umowy oraz miesięczny czynsz.

**Opis utworzonych widoków:**

1. **PrzegladTransakcjiAktywne:** Widok ten przedstawia szczegóły aktywnych transakcji nieruchomości, włącznie z danymi takimi jak adres nieruchomości, klient, agent, data transakcji, kwota oraz prowizja agenta, jeśli istnieje.
2. **AktywniKlienciWMiastach:** Ten widok prezentuje liczbę aktywnych klientów w poszczególnych miastach, gdzie aktywność klienta jest określona przez zdefiniowany status aktywny w tabeli klientów.
3. **StatystykiTransakcjiMiesieczne:** Widok zbiera statystyki miesięczne dotyczące transakcji, takie jak liczba transakcji, suma kwot transakcji oraz średnia kwota transakcji, grupując je według roku i miesiąca.
4. **SredniaCenaNieruchomosciDrogichWMiastach:** Ten widok oblicza średnią cenę nieruchomości powyżej określonej wartości (tutaj 500,000) dla różnych miast, uwzględniając jedynie te nieruchomości, które spełniają kryterium cenowe.
5. **AktywnoscAgentow:** Widok przedstawia aktywność agentów w sprzedaży nieruchomości, licząc ilość transakcji, sumę kwot transakcji oraz sumę prowizji dla każdego z agentów.
6. **SredniaCenaNieruchomosciWMiastach:** Widok ten pokazuje średnią cenę nieruchomości dla różnych miast, niezależnie od wartości czy statusu.
7. **LiczbaKlientowWMiastach:** Widok ten prezentuje liczbę klientów w poszczególnych miastach, grupując ich według miast z tabeli miast i klientów.



Kod:

-- Sprawdzenie, czy baza danych istnieje i jej usunięcie, jeśli istnieje

IF DB\_ID('FirmaNieruchomosci') IS NOT NULL

BEGIN

DROP DATABASE FirmaNieruchomosci;

END

GO

-- Utworzenie bazy danych

CREATE DATABASE FirmaNieruchomosci;

GO

USE FirmaNieruchomosci;

GO

-- Tabela Miasta

CREATE TABLE Miasta (

ID\_Miasta INT PRIMARY KEY IDENTITY(1,1),

NazwaMiasta NVARCHAR(100) NOT NULL,

Kraj NVARCHAR(50) NOT NULL,

LiczbaMieszkancow INT CHECK (LiczbaMieszkancow > 0)

);

CREATE INDEX idx\_NazwaMiasta ON Miasta(NazwaMiasta);

GO

-- Tabela TypyNieruchomosci

CREATE TABLE TypyNieruchomosci (

ID\_Typu INT PRIMARY KEY IDENTITY(1,1),

NazwaTypu NVARCHAR(50) NOT NULL,

Opis NVARCHAR(255),

Aktywny BIT DEFAULT 1

);

GO

-- Tabela StatusyTransakcji

CREATE TABLE StatusyTransakcji (

ID\_Statusu INT PRIMARY KEY IDENTITY(1,1),

NazwaStatusu NVARCHAR(50) NOT NULL,

Opis NVARCHAR(255),

DataDodania DATE DEFAULT GETDATE()

);

GO

-- Tabela Nieruchomosci

CREATE TABLE Nieruchomosci (

ID\_Nieruchomosci INT PRIMARY KEY IDENTITY(1,1),

Adres NVARCHAR(255) NOT NULL,

MiastoID INT NOT NULL,

Cena DECIMAL(18,2) CHECK (Cena > 0),

TypID INT NOT NULL,

Powierzchnia INT CHECK (Powierzchnia > 0),

DataDodania DATE,

StatusID INT NOT NULL,

FOREIGN KEY (MiastoID) REFERENCES Miasta(ID\_Miasta),

FOREIGN KEY (TypID) REFERENCES TypyNieruchomosci(ID\_Typu),

FOREIGN KEY (StatusID) REFERENCES StatusyTransakcji(ID\_Statusu)

);

CREATE INDEX idx\_ID\_Nieruchomosci ON Nieruchomosci(ID\_Nieruchomosci);

CREATE INDEX idx\_Nieruchomosci\_MiastoID ON Nieruchomosci(MiastoID);

CREATE INDEX idx\_Nieruchomosci\_TypID ON Nieruchomosci(TypID);

CREATE INDEX idx\_Nieruchomosci\_StatusID ON Nieruchomosci(StatusID);

GO

-- Tabela Klienci

CREATE TABLE Klienci (

ID\_Klienta INT PRIMARY KEY IDENTITY(1,1),

Imie NVARCHAR(50) NOT NULL,

Nazwisko NVARCHAR(50) NOT NULL,

Email NVARCHAR(100) CHECK (Email LIKE '%@%'),

Telefon NVARCHAR(20),

MiastoID INT NOT NULL,

FOREIGN KEY (MiastoID) REFERENCES Miasta(ID\_Miasta)

);

CREATE INDEX idx\_Klienci\_MiastoID ON Klienci(MiastoID);

GO

-- Tabela Agenci

CREATE TABLE Agenci (

ID\_Agenta INT PRIMARY KEY IDENTITY(1,1),

Imie NVARCHAR(50) NOT NULL,

Nazwisko NVARCHAR(50) NOT NULL,

Email NVARCHAR(100),

Telefon NVARCHAR(20),

MiastoID INT NOT NULL,

FOREIGN KEY (MiastoID) REFERENCES Miasta(ID\_Miasta)

);

CREATE INDEX idx\_Agenci\_MiastoID ON Agenci(MiastoID);

GO

-- Tabela Transakcje

CREATE TABLE Transakcje (

ID\_Transakcji INT PRIMARY KEY IDENTITY(1,1),

ID\_Nieruchomosci INT NOT NULL,

ID\_Klienta INT NOT NULL,

ID\_Agenta INT NOT NULL,

DataTransakcji DATE,

Kwota DECIMAL(18,2) CHECK (Kwota > 0),

StatusID INT NOT NULL,

FOREIGN KEY (ID\_Nieruchomosci) REFERENCES Nieruchomosci(ID\_Nieruchomosci),

FOREIGN KEY (ID\_Klienta) REFERENCES Klienci(ID\_Klienta),

FOREIGN KEY (ID\_Agenta) REFERENCES Agenci(ID\_Agenta),

FOREIGN KEY (StatusID) REFERENCES StatusyTransakcji(ID\_Statusu)

);

CREATE INDEX idx\_ID\_Transakcji ON Transakcje(ID\_Transakcji);

CREATE INDEX idx\_Transakcje\_ID\_Nieruchomosci ON Transakcje(ID\_Nieruchomosci);

CREATE INDEX idx\_Transakcje\_ID\_Klienta ON Transakcje(ID\_Klienta);

CREATE INDEX idx\_Transakcje\_ID\_Agenta ON Transakcje(ID\_Agenta);

CREATE INDEX idx\_Transakcje\_StatusID ON Transakcje(StatusID);

GO

-- Tabela ProwizjeAgentow

CREATE TABLE ProwizjeAgentow (

ID\_Prowizji INT PRIMARY KEY IDENTITY(1,1),

ID\_Agenta INT NOT NULL,

ID\_Transakcji INT NOT NULL,

Prowizja DECIMAL(18,2) CHECK (Prowizja >= 0),

DataWyplaty DATE,

FOREIGN KEY (ID\_Agenta) REFERENCES Agenci(ID\_Agenta),

FOREIGN KEY (ID\_Transakcji) REFERENCES Transakcje(ID\_Transakcji)

);

GO

-- Tabela Deweloperzy

CREATE TABLE Deweloperzy (

ID\_Dewelopera INT PRIMARY KEY IDENTITY(1,1),

Nazwa NVARCHAR(100) NOT NULL,

Adres NVARCHAR(255),

Email NVARCHAR(100) CHECK (Email LIKE '%@%'),

Telefon NVARCHAR(20) CHECK (Telefon LIKE '[0-9]%'),

MiastoID INT NOT NULL,

FOREIGN KEY (MiastoID) REFERENCES Miasta(ID\_Miasta)

);

GO

-- Tabela UmowyNajmu

CREATE TABLE UmowyNajmu (

ID\_Umowy INT PRIMARY KEY IDENTITY(1,1),

ID\_Nieruchomosci INT NOT NULL,

ID\_Klienta INT NOT NULL,

DataRozpoczecia DATE NOT NULL,

DataZakonczenia DATE NOT NULL,

MiesiecznyCzynsz DECIMAL(18,2) CHECK (MiesiecznyCzynsz > 0),

FOREIGN KEY (ID\_Nieruchomosci) REFERENCES Nieruchomosci(ID\_Nieruchomosci),

FOREIGN KEY (ID\_Klienta) REFERENCES Klienci(ID\_Klienta)

);

GO

-- Wstawienie przykładowych danych

-- Miasta

INSERT INTO Miasta (NazwaMiasta, Kraj, LiczbaMieszkancow) VALUES ('Warszawa', 'Polska', 1790658);

INSERT INTO Miasta (NazwaMiasta, Kraj, LiczbaMieszkancow) VALUES ('Kraków', 'Polska', 779115);

INSERT INTO Miasta (NazwaMiasta, Kraj, LiczbaMieszkancow) VALUES ('Wrocław', 'Polska', 643782);

-- TypyNieruchomosci

INSERT INTO TypyNieruchomosci (NazwaTypu, Opis) VALUES ('Mieszkanie', 'Lokal mieszkalny w bloku lub kamienicy');

INSERT INTO TypyNieruchomosci (NazwaTypu, Opis) VALUES ('Dom', 'Wolnostojący budynek mieszkalny');

INSERT INTO TypyNieruchomosci (NazwaTypu, Opis) VALUES ('Działka', 'Grunt pod zabudowę');

INSERT INTO TypyNieruchomosci (NazwaTypu, Opis) VALUES ('Biuro', 'Lokal przeznaczony do pracy biurowej');

-- StatusyTransakcji

INSERT INTO StatusyTransakcji (NazwaStatusu, Opis) VALUES ('W trakcie', 'Transakcja jest w trakcie realizacji');

INSERT INTO StatusyTransakcji (NazwaStatusu, Opis) VALUES ('Zakończona', 'Transakcja została zakończona pomyślnie');

INSERT INTO StatusyTransakcji (NazwaStatusu, Opis) VALUES ('Anulowana', 'Transakcja została anulowana');

-- Nieruchomosci

INSERT INTO Nieruchomosci (Adres, MiastoID, Cena, TypID, Powierzchnia, DataDodania, StatusID) VALUES ('ul. Marszałkowska 1', 1, 850000.00, 1, 60, '2023-01-15', 1);

INSERT INTO Nieruchomosci (Adres, MiastoID, Cena, TypID, Powierzchnia, DataDodania, StatusID) VALUES ('ul. Floriańska 10', 2, 1250000.00, 2, 120, '2023-02-20', 2);

INSERT INTO Nieruchomosci (Adres, MiastoID, Cena, TypID, Powierzchnia, DataDodania, StatusID) VALUES ('ul. Rynek 5', 3, 400000.00, 3, 300, '2023-03-10', 3);

-- Klienci

INSERT INTO Klienci (Imie, Nazwisko, Email, Telefon, MiastoID) VALUES ('Jan', 'Kowalski', 'jan.kowalski@example.com', '123456789', 1);

INSERT INTO Klienci (Imie, Nazwisko, Email, Telefon, MiastoID) VALUES ('Anna', 'Nowak', 'anna.nowak@example.com', '987654321', 2);

INSERT INTO Klienci (Imie, Nazwisko, Email, Telefon, MiastoID) VALUES ('Piotr', 'Wiśniewski', 'piotr.wisniewski@example.com', '567890123', 3);

-- Agenci

INSERT INTO Agenci (Imie, Nazwisko, Email, Telefon, MiastoID) VALUES ('Katarzyna', 'Mazur', 'katarzyna.mazur@example.com', '111222333', 1);

INSERT INTO Agenci (Imie, Nazwisko, Email, Telefon, MiastoID) VALUES ('Michał', 'Dąbrowski', 'michal.dabrowski@example.com', '444555666', 2);

INSERT INTO Agenci (Imie, Nazwisko, Email, Telefon, MiastoID) VALUES ('Monika', 'Zielińska', 'monika.zielinska@example.com', '777888999', 3);

-- Transakcje

INSERT INTO Transakcje (ID\_Nieruchomosci, ID\_Klienta, ID\_Agenta, DataTransakcji, Kwota, StatusID) VALUES (1, 1, 1, '2023-05-01', 850000.00, 2);

INSERT INTO Transakcje (ID\_Nieruchomosci, ID\_Klienta, ID\_Agenta, DataTransakcji, Kwota, StatusID) VALUES (2, 2, 2, '2023-06-15', 1250000.00, 2);

INSERT INTO Transakcje (ID\_Nieruchomosci, ID\_Klienta, ID\_Agenta, DataTransakcji, Kwota, StatusID) VALUES (3, 3, 3, '2023-07-20', 400000.00, 2);

-- ProwizjeAgentow

INSERT INTO ProwizjeAgentow (ID\_Agenta, ID\_Transakcji, Prowizja, DataWyplaty) VALUES (1, 1, 8500.00, '2023-05-10');

INSERT INTO ProwizjeAgentow (ID\_Agenta, ID\_Transakcji, Prowizja, DataWyplaty) VALUES (2, 2, 12500.00, '2023-06-20');

INSERT INTO ProwizjeAgentow (ID\_Agenta, ID\_Transakcji, Prowizja, DataWyplaty) VALUES (3, 3, 4000.00, '2023-07-25');

-- Deweloperzy

INSERT INTO Deweloperzy (Nazwa, Adres, Email, Telefon, MiastoID) VALUES ('Budimex', 'ul. Przykładowa 1, Warszawa', 'kontakt@budimex.com', '222333444', 1);

INSERT INTO Deweloperzy (Nazwa, Adres, Email, Telefon, MiastoID) VALUES ('Echo Investment', 'ul. Przykładowa 2, Kraków', 'kontakt@echo.com', '555666777', 2);

INSERT INTO Deweloperzy (Nazwa, Adres, Email, Telefon, MiastoID) VALUES ('Skanska', 'ul. Przykładowa 3, Wrocław', 'kontakt@skanska.com', '888999000', 3);

-- UmowyNajmu

INSERT INTO UmowyNajmu (ID\_Nieruchomosci, ID\_Klienta, DataRozpoczecia, DataZakonczenia, MiesiecznyCzynsz) VALUES (1, 1, '2023-08-01', '2024-08-01', 2500.00);

INSERT INTO UmowyNajmu (ID\_Nieruchomosci, ID\_Klienta, DataRozpoczecia, DataZakonczenia, MiesiecznyCzynsz) VALUES (2, 2, '2023-09-01', '2024-09-01', 3500.00);

INSERT INTO UmowyNajmu (ID\_Nieruchomosci, ID\_Klienta, DataRozpoczecia, DataZakonczenia, MiesiecznyCzynsz) VALUES (3, 3, '2023-10-01', '2024-10-01', 1500.00);

GO

-- Tworzenie widoków

-- Widok 1: Przegląd transakcji z prowizjami dla aktywnych transakcji

IF OBJECT\_ID('dbo.PrzegladTransakcjiAktywne', 'V') IS NOT NULL

DROP VIEW dbo.PrzegladTransakcjiAktywne;

GO

CREATE VIEW PrzegladTransakcjiAktywne

AS

SELECT

T.ID\_Transakcji,

N.Adres AS AdresNieruchomosci,

K.Imie + ' ' + K.Nazwisko AS Klient,

A.Imie + ' ' + A.Nazwisko AS Agent,

T.DataTransakcji,

T.Kwota,

PT.Prowizja

FROM Transakcje T

INNER JOIN Nieruchomosci N ON T.ID\_Nieruchomosci = N.ID\_Nieruchomosci

INNER JOIN Klienci K ON T.ID\_Klienta = K.ID\_Klienta

INNER JOIN Agenci A ON T.ID\_Agenta = A.ID\_Agenta

LEFT JOIN ProwizjeAgentow PT ON T.ID\_Transakcji = PT.ID\_Transakcji

WHERE T.StatusID = 2; -- Aktywne transakcje

GO

-- Widok 2: Liczba klientów w poszczególnych miastach dla aktywnych klientów

IF OBJECT\_ID('dbo.AktywniKlienciWMiastach', 'V') IS NOT NULL

DROP VIEW dbo.AktywniKlienciWMiastach;

GO

CREATE VIEW AktywniKlienciWMiastach

AS

SELECT

M.NazwaMiasta,

COUNT(K.ID\_Klienta) AS LiczbaKlientow

FROM Klienci K

INNER JOIN Miasta M ON K.MiastoID = M.ID\_Miasta

GROUP BY M.NazwaMiasta;

GO

-- Widok 3: Statystyki transakcji miesięczne

IF OBJECT\_ID('dbo.StatystykiTransakcjiMiesieczne', 'V') IS NOT NULL

DROP VIEW dbo.StatystykiTransakcjiMiesieczne;

GO

CREATE VIEW StatystykiTransakcjiMiesieczne

AS

SELECT

YEAR(T.DataTransakcji) AS Rok,

MONTH(T.DataTransakcji) AS Miesiac,

COUNT(T.ID\_Transakcji) AS LiczbaTransakcji,

SUM(T.Kwota) AS SumaKwot,

CAST(ROUND(AVG(T.Kwota), 2) AS DECIMAL(18,2)) AS SredniaKwota

FROM Transakcje T

GROUP BY YEAR(T.DataTransakcji), MONTH(T.DataTransakcji);

GO

-- Widok 4: Średnia cena nieruchomości w różnych miastach dla nieruchomości powyżej określonej wartości

IF OBJECT\_ID('dbo.SredniaCenaNieruchomosciDrogichWMiastach', 'V') IS NOT NULL

DROP VIEW dbo.SredniaCenaNieruchomosciDrogichWMiastach;

GO

CREATE VIEW SredniaCenaNieruchomosciDrogichWMiastach

AS

SELECT

M.NazwaMiasta,

CAST(ROUND(AVG(N.Cena), 2) AS DECIMAL(18,2)) AS SredniaCena

FROM Nieruchomosci N

INNER JOIN Miasta M ON N.MiastoID = M.ID\_Miasta

WHERE N.Cena > 500000 -- Cena nieruchomości powyżej 500,000

GROUP BY M.NazwaMiasta;

GO

-- Widok 5: Aktywność agentów w sprzedaży nieruchomości

IF OBJECT\_ID('dbo.AktywnoscAgentow', 'V') IS NOT NULL

DROP VIEW dbo.AktywnoscAgentow;

GO

CREATE VIEW AktywnoscAgentow

AS

SELECT

A.ID\_Agenta,

A.Imie + ' ' + A.Nazwisko AS Agent,

COUNT(T.ID\_Transakcji) AS LiczbaTransakcji,

SUM(T.Kwota) AS SumaKwot,

SUM(PT.Prowizja) AS SumaProwizji

FROM Agenci A

LEFT JOIN Transakcje T ON A.ID\_Agenta = T.ID\_Agenta

LEFT JOIN ProwizjeAgentow PT ON T.ID\_Transakcji = PT.ID\_Transakcji

GROUP BY A.ID\_Agenta, A.Imie, A.Nazwisko;

GO

-- Widok 6: Średnia cena nieruchomości w różnych miastach

IF OBJECT\_ID('dbo.SredniaCenaNieruchomosciWMiastach', 'V') IS NOT NULL

DROP VIEW dbo.SredniaCenaNieruchomosciWMiastach;

GO

CREATE VIEW SredniaCenaNieruchomosciWMiastach

AS

SELECT

M.NazwaMiasta,

CAST(ROUND(AVG(N.Cena), 2) AS DECIMAL(18,2)) AS SredniaCena

FROM Nieruchomosci N

INNER JOIN Miasta M ON N.MiastoID = M.ID\_Miasta

GROUP BY M.NazwaMiasta;

GO

-- Widok 7: Liczba klientów w poszczególnych miastach

IF OBJECT\_ID('dbo.LiczbaKlientowWMiastach', 'V') IS NOT NULL

DROP VIEW dbo.LiczbaKlientowWMiastach;

GO

CREATE VIEW LiczbaKlientowWMiastach

AS

SELECT

M.NazwaMiasta,

COUNT(K.ID\_Klienta) AS LiczbaKlientow

FROM Klienci K

INNER JOIN Miasta M ON K.MiastoID = M.ID\_Miasta

GROUP BY M.NazwaMiasta;

GO

-- Wywołanie widoku PrzegladTransakcjiAktywne

SELECT \* FROM PrzegladTransakcjiAktywne;

-- Wywołanie widoku AktywniKlienciWMiastach

SELECT \* FROM AktywniKlienciWMiastach;

-- Wywołanie widoku StatystykiTransakcjiMiesieczne

SELECT \* FROM StatystykiTransakcjiMiesieczne;

-- Wywołanie widoku SredniaCenaNieruchomosciDrogichWMiastach

SELECT \* FROM SredniaCenaNieruchomosciDrogichWMiastach;

-- Wywołanie widoku AktywnoscAgentow

SELECT \* FROM AktywnoscAgentow;

-- Wywołanie widoku SredniaCenaNieruchomosciWMiastach

SELECT \* FROM SredniaCenaNieruchomosciWMiastach;

-- Wywołanie widoku LiczbaKlientowWMiastach

SELECT \* FROM LiczbaKlientowWMiastach;

-- Tworzenie funkcji

-- Funkcja 1: Znajdź transakcje klienta

IF OBJECT\_ID('dbo.ZnajdzTransakcjeKlienta', 'TF') IS NOT NULL

DROP FUNCTION dbo.ZnajdzTransakcjeKlienta;

GO

CREATE FUNCTION ZnajdzTransakcjeKlienta

(

@ID\_Klienta INT

)

RETURNS TABLE

AS

RETURN

(

SELECT

T.ID\_Transakcji,

N.Adres AS AdresNieruchomosci,

T.DataTransakcji,

T.Kwota,

PT.Prowizja

FROM Transakcje T

INNER JOIN Nieruchomosci N ON T.ID\_Nieruchomosci = N.ID\_Nieruchomosci

LEFT JOIN ProwizjeAgentow PT ON T.ID\_Transakcji = PT.ID\_Transakcji

WHERE T.ID\_Klienta = @ID\_Klienta

);

GO

-- Funkcja 2: Znajdź umowy najmu agenta

IF OBJECT\_ID('dbo.ZnajdzUmowyNajmuAgenta', 'TF') IS NOT NULL

DROP FUNCTION dbo.ZnajdzUmowyNajmuAgenta;

GO

CREATE FUNCTION ZnajdzUmowyNajmuAgenta

(

@ID\_Agenta INT

)

RETURNS TABLE

AS

RETURN

(

SELECT

UN.ID\_Umowy,

N.Adres AS AdresNieruchomosci,

UN.DataRozpoczecia,

UN.DataZakonczenia,

UN.MiesiecznyCzynsz

FROM UmowyNajmu UN

INNER JOIN Transakcje T ON UN.ID\_Nieruchomosci = T.ID\_Nieruchomosci

INNER JOIN Agenci A ON T.ID\_Agenta = A.ID\_Agenta

INNER JOIN Nieruchomosci N ON UN.ID\_Nieruchomosci = N.ID\_Nieruchomosci

WHERE A.ID\_Agenta = @ID\_Agenta

);

GO

-- Funkcja 3: Znajdź ostatnie transakcje

IF OBJECT\_ID('dbo.ZnajdzOstatnieTransakcje', 'TF') IS NOT NULL

DROP FUNCTION dbo.ZnajdzOstatnieTransakcje;

GO

CREATE FUNCTION ZnajdzOstatnieTransakcje

()

RETURNS TABLE

AS

RETURN

(

SELECT DISTINCT

K.ID\_Klienta,

MAX(T.DataTransakcji) AS OstatniaTransakcja

FROM Klienci K

INNER JOIN Transakcje T ON K.ID\_Klienta = T.ID\_Klienta

GROUP BY K.ID\_Klienta

);

GO

-- Funkcja 4: Znajdź klientów z największą liczbą transakcji

IF OBJECT\_ID('dbo.ZnajdzKlientowNajwiecejTransakcji', 'TF') IS NOT NULL

DROP FUNCTION dbo.ZnajdzKlientowNajwiecejTransakcji;

GO

CREATE FUNCTION ZnajdzKlientowNajwiecejTransakcji

()

RETURNS TABLE

AS

RETURN

(

SELECT TOP 10

K.ID\_Klienta,

K.Imie,

K.Nazwisko,

COUNT(T.ID\_Transakcji) AS LiczbaTransakcji

FROM Klienci K

LEFT JOIN Transakcje T ON K.ID\_Klienta = T.ID\_Klienta

GROUP BY K.ID\_Klienta, K.Imie, K.Nazwisko

ORDER BY LiczbaTransakcji DESC

);

GO

-- Wywołanie funkcji w celu sprawdzenia działania

-- 1. Znajdź transakcje klienta (przykładowo dla klienta o ID\_Klienta = 1)

SELECT \*

FROM dbo.ZnajdzTransakcjeKlienta(1);

-- 2. Znajdź umowy najmu agenta (przykładowo dla agenta o ID\_Agenta = 2)

SELECT \*

FROM dbo.ZnajdzUmowyNajmuAgenta(2);

-- 3. Znajdź ostatnie transakcje

SELECT \*

FROM dbo.ZnajdzOstatnieTransakcje();

-- 4. Znajdź klientów z największą liczbą transakcji

SELECT \*

FROM dbo.ZnajdzKlientowNajwiecejTransakcji();

-- Tworzenie procedur składowanych

-- Procedura 1: Zaktualizuj datę rozpoczęcia umowy najmu tylko jeśli umowa istnieje

IF OBJECT\_ID('dbo.ZaktualizujDateRozpoczeciaUmowyNajmu', 'P') IS NOT NULL

DROP PROCEDURE dbo.ZaktualizujDateRozpoczeciaUmowyNajmu;

GO

CREATE PROCEDURE ZaktualizujDateRozpoczeciaUmowyNajmu

@ID\_Umowy INT,

@NowaDataRozpoczecia DATE

AS

BEGIN

DECLARE @LiczbaUmow INT;

SELECT @LiczbaUmow = COUNT(\*)

FROM UmowyNajmu

WHERE ID\_Umowy = @ID\_Umowy;

IF @LiczbaUmow > 0

BEGIN

UPDATE UmowyNajmu

SET DataRozpoczecia = @NowaDataRozpoczecia

WHERE ID\_Umowy = @ID\_Umowy;

PRINT 'Data rozpoczęcia umowy najmu została zaktualizowana.';

END

ELSE

BEGIN

PRINT 'Umowa o podanym ID nie istnieje.';

END

END;

GO

-- Procedura 2: Dodaj nowego klienta

IF OBJECT\_ID('dbo.DodajNowegoKlienta', 'P') IS NOT NULL

DROP PROCEDURE dbo.DodajNowegoKlienta;

GO

CREATE PROCEDURE DodajNowegoKlienta

@Imie NVARCHAR(50),

@Nazwisko NVARCHAR(50),

@Email NVARCHAR(100),

@Telefon NVARCHAR(20),

@MiastoID INT

AS

BEGIN

IF LEN(@Imie) > 0 AND LEN(@Nazwisko) > 0 AND LEN(@Email) > 0 AND LEN(@Telefon) > 0

BEGIN

-- Sprawdź, czy klient o podanym adresie email już istnieje

IF NOT EXISTS (SELECT 1 FROM Klienci WHERE Email = @Email)

BEGIN

-- Dodaj nowego klienta, jeśli nie istnieje klient o tym adresie email

INSERT INTO Klienci (Imie, Nazwisko, Email, Telefon, MiastoID)

VALUES (@Imie, @Nazwisko, @Email, @Telefon, @MiastoID);

PRINT 'Nowy klient został dodany.';

END

ELSE

BEGIN

PRINT 'Nie udało się dodać nowego klienta. Klient o podanym adresie email już istnieje.';

END

END

ELSE

BEGIN

PRINT 'Nie udało się dodać nowego klienta. Wymagane są wszystkie dane: Imię, Nazwisko, Email, Telefon.';

END

END;

GO

-- Procedura 3: Aktualizacja ceny nieruchomości

IF OBJECT\_ID('dbo.AktualizujCeneNieruchomosci', 'P') IS NOT NULL

DROP PROCEDURE dbo.AktualizujCeneNieruchomosci;

GO

CREATE PROCEDURE AktualizujCeneNieruchomosci

@ID\_Nieruchomosci INT,

@NowaCena MONEY

AS

BEGIN

DECLARE @StatusAktualny INT;

-- Sprawdź status aktualny nieruchomości

SELECT @StatusAktualny = StatusID

FROM Nieruchomosci

WHERE ID\_Nieruchomosci = @ID\_Nieruchomosci;

-- Jeśli nieruchomość ma status "Aktywna"

IF @StatusAktualny = 1

BEGIN

-- Zaktualizuj cenę nieruchomości

UPDATE Nieruchomosci

SET Cena = @NowaCena

WHERE ID\_Nieruchomosci = @ID\_Nieruchomosci;

PRINT 'Cena nieruchomości została zaktualizowana.';

END

ELSE

BEGIN

PRINT 'Nie można zaktualizować ceny nieruchomości. Nieruchomość nie jest w aktywnym stanie.';

END

END;

GO

-- Procedura 4: Znajdź klientów z największą liczbą transakcji

IF OBJECT\_ID('dbo.Znajdz\_KlientowNajwiecejTransakcji', 'P') IS NOT NULL

DROP PROCEDURE dbo.Znajdz\_KlientowNajwiecejTransakcji;

GO

CREATE PROCEDURE Znajdz\_KlientowNajwiecejTransakcji

AS

BEGIN

SELECT TOP 10

K.ID\_Klienta,

K.Imie,

K.Nazwisko,

COUNT(T.ID\_Transakcji) AS LiczbaTransakcji

FROM Klienci K

LEFT JOIN Transakcje T ON K.ID\_Klienta = T.ID\_Klienta

GROUP BY K.ID\_Klienta, K.Imie, K.Nazwisko

ORDER BY LiczbaTransakcji DESC;

END;

GO

-- Przykłady wywołania procedur składowanych:

-- Wywołanie procedury 1: Zaktualizuj datę rozpoczęcia umowy najmu

EXEC ZaktualizujDateRozpoczeciaUmowyNajmu @ID\_Umowy = 1, @NowaDataRozpoczecia = '2024-06-30';

-- Wywołanie procedury 2: Dodaj nowego klienta

EXEC DodajNowegoKlienta @Imie = 'Jan', @Nazwisko = 'Kowalski', @Email = 'jan.kowalski@example.com', @Telefon = '123456789', @MiastoID = 3;

-- Wywołanie procedury 3: Aktualizacja ceny nieruchomości

EXEC AktualizujCeneNieruchomosci @ID\_Nieruchomosci = 2, @NowaCena = 350000;

-- Wywołanie procedury 4: Znajdź klientów z największą liczbą transakcji

EXEC Znajdz\_KlientowNajwiecejTransakcji;

-- Tworzenie wyzwalaczy

-- Wyzwalacz 1: Wyzwalacz do sprawdzania poprawności danych w tabeli Klienci

IF OBJECT\_ID('SprawdzPoprawnoscDanychKlientow', 'TR') IS NOT NULL

DROP TRIGGER SprawdzPoprawnoscDanychKlientow;

GO

CREATE TRIGGER SprawdzPoprawnoscDanychKlientow

ON Klienci

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT 1 FROM inserted WHERE Email NOT LIKE '%@%')

BEGIN

RAISERROR ('Niepoprawny format adresu email.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

IF EXISTS (SELECT 1 FROM inserted WHERE NOT Telefon LIKE '[0-9]%')

BEGIN

RAISERROR ('Niepoprawny format numeru telefonu.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

END;

GO

-- Wyzwalacz 2: Wyzwalacz przed wstawieniem nowego agenta

IF OBJECT\_ID('WstawianieNowegoAgenta', 'TR') IS NOT NULL

DROP TRIGGER WstawianieNowegoAgenta;

GO

CREATE TRIGGER WstawianieNowegoAgenta

ON Agenci

INSTEAD OF INSERT

AS

BEGIN

DECLARE @Imie NVARCHAR(50), @Nazwisko NVARCHAR(50), @MiastoID INT;

SELECT @Imie = Imie, @Nazwisko = Nazwisko, @MiastoID = MiastoID FROM inserted;

IF LEN(@Imie) > 1 AND LEN(@Nazwisko) > 1

BEGIN

INSERT INTO Agenci (Imie, Nazwisko, MiastoID)

VALUES (@Imie, @Nazwisko, @MiastoID);

PRINT 'Nowy agent został dodany.';

END

ELSE

BEGIN

PRINT 'Nie udało się dodać nowego agenta. Imię i nazwisko muszą mieć co najmniej 2 znaki.';

END

END;

GO

-- Testowanie wyzwalaczy

-- Test SprawdzPoprawnoscDanychKlientow

PRINT 'Testowanie wyzwalacza SprawdzPoprawnoscDanychKlientow';

BEGIN TRY

INSERT INTO Klienci (Imie, Nazwisko, Email, Telefon, MiastoID)

VALUES ('Jan', 'Kowalski', 'jan.kowalski@example.com', '123456789', 1); -- Zakładając, że MiastoID = 1 jest poprawne

PRINT 'Poprawne dane dodane.';

END TRY

BEGIN CATCH

PRINT 'Błąd: ' + ERROR\_MESSAGE();

END CATCH

BEGIN TRY

INSERT INTO Klienci (Imie, Nazwisko, Email, Telefon, MiastoID)

VALUES ('Piotr', 'Zieliński', 'piotr.zielinski@example.com', 'ABC123', 1);

PRINT 'Błąd: dane nie powinny zostać dodane z niepoprawnym numerem telefonu.';

END TRY

BEGIN CATCH

PRINT 'Błąd: ' + ERROR\_MESSAGE();

END CATCH

-- Test WstawianieNowegoAgenta

PRINT 'Testowanie wyzwalacza WstawianieNowegoAgenta';

BEGIN TRY

INSERT INTO Agenci (Imie, Nazwisko, MiastoID)

VALUES ('Jan', 'Kowalski',1);

PRINT 'Nowy agent został dodany.';

END TRY

BEGIN CATCH

PRINT 'Błąd: ' + ERROR\_MESSAGE();

END CATCH

BEGIN TRY

INSERT INTO Agenci (Imie, Nazwisko) VALUES ('A', 'B');

PRINT 'Błąd: dane nie powinny zostać dodane z imieniem i nazwiskiem krótszym niż 2 znaki.';

END TRY

BEGIN CATCH

PRINT 'Błąd: ' + ERROR\_MESSAGE();

END CATCH