Baza Danych Pogotowia Ratunkowego

Projekt bazy danych

Aleksandra Grygorczyk
Ignacy Kolton
2022/2023

Spis Treści

2. Srednia i Max kosztow dziennych
3. Statystyki Zespołów
4. Skład Zespołów
5. Wyjazdy dzienne Statystyki
6. Zgloszenia bez wyjazdu
7. Zgloszenia a wyjazdy dzienne

Funkcje:

1. Sprawdzenie Daty Urodzenia
2. Sprawdzenie czy nie ma 2 urlopow na raz
3. Sprawdzenie nr PESEL

Procedury składowane:

1. Informacje o wyjeździe karetki
2. Wypłata za okres dla Pracownika
3. Kiedy ostatni raz został zakupiony ten przedmiot i w jakiej cenie
4. Ile osób jest w danym zespole
5. Ile godzin przepracował pracownik w danym okresie
6. Grafik zespołów na konkretny dzień

Wyzwalacze:

1. Suma w Zakupy wyposażenia
2. Najpóźniejsza data następnego przeglądu karetki
3. Sprawdzanie czy wpisana stawka jest nieujemna
4. Automatyczny zapis daty w tabeli Zgłoszenia
5. Byli pracownicy

Założenia Projektowe

Projekt bazy danych pogotowia ratunkowego ma na celu stworzenie skutecznego i efektywnego systemu przechowywania, udostępniania i analizowania danych dotyczących działalności pogotowia ratunkowego. Baza danych będzie zawierać informacje o pacjentach, interwencjach, sprzęcie i personelu, a także umożliwi analizowanie danych i podsumowywanie działalności. Dzięki temu, baza danych pogotowia ratunkowego pozwoli na lepsze planowanie i koordynację działań pogotowia, a także na poprawę jakości opieki medycznej udzielanej pacjentom.

Baza danych pogotowia ratunkowego będzie składać się z trzech modułów, które będą odpowiedzialne za różne funkcjonalności.

1. **Moduł zarządzania interwencjami** pozwoli na planowanie i koordynację działań zespołów ratunkowych oraz śledzenie postępów interwencji.
2. **Moduł zarządzania sprzętem i kosztami** pozwoli na monitorowanie stanu sprzętu oraz planowanie przeglądów i napraw.
3. **Modul zarządzania personelem** pozwoli na planowanie grafików, urlopów oraz obliczanie wypłat.

Strategia Pielęgnacji bazy danych

1. ***Regularne aktualizacje i konserwacja bazy danych.***

Comiesięczne aktualizacje obejmują zmiany w oprogramowaniu bazy danych, aktualizacje bezpieczeństwa, poprawki błędów oraz nowe funkcjonalności. Usuwanie niepotrzebnych danych, reorganizacja tabel, defragmentacja i optymalizacja indeksów, powinna odbywać się co tydzień lub w miarę potrzeb, co poprawi wydajność bazy danych.

2. ***Regularne tworzenie kopii zapasowych.***

Backup powinien być tworzony codziennie. Kopie zapasowe będą przechowywane w bezpiecznym miejscu, takim jak dysk zewnętrzny lub chmura.

Aby upewnić się, że proces tworzenia kopii zapasowych jest skuteczny, należy raz w miesiącu testować proces odzyskiwania danych.

Pracownicy odpowiedzialni za działanie w przypadku awarii bazy danych powinni być odpowiednio przeszkoleni, aby byli przygotowani do działania.

Plan działania w razie awarii:

- Identyfikacja awarii.
- Następnie należy podjąć szybkie działania mające na celu minimalizację utraty danych. Czyli odtworzyć ostatnią zapisaną kopię i przywrócić działanie serwera bazy danych.
- W trakcie procesu odzyskiwania należy zabezpieczyć dane przed dalszym uszkodzeniem, np. poprzez szyfrowanie danych lub utworzenie kopii zapasowej.

3. ***Audyt i testowanie:***

Audyt i testowanie bazy danych pogotowia ratunkowego w celu upewnienia się, że spełnia ona wymagania i jest zgodna z przepisami.

Coroczny audyt polega na przeglądaniu struktury bazy, danych oraz procesów związanych z jej zarządzaniem, celem wykrycia nieprawidłowości lub braków. Audyt może być przeprowadzany przez wewnętrzny zespół lub zewnętrzny podmiot specjalizujący się w tego typu działaniach.

Diagram ER

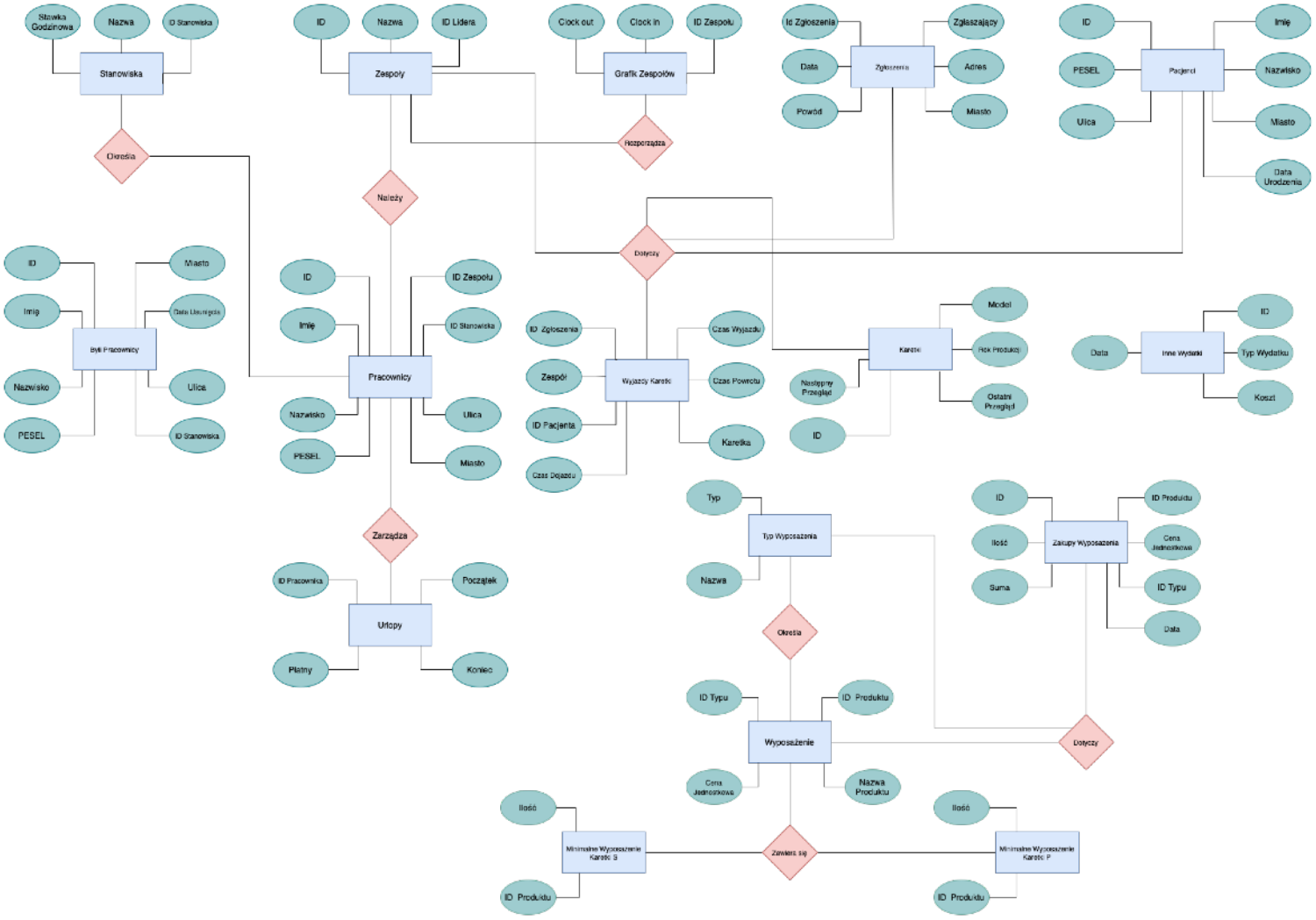
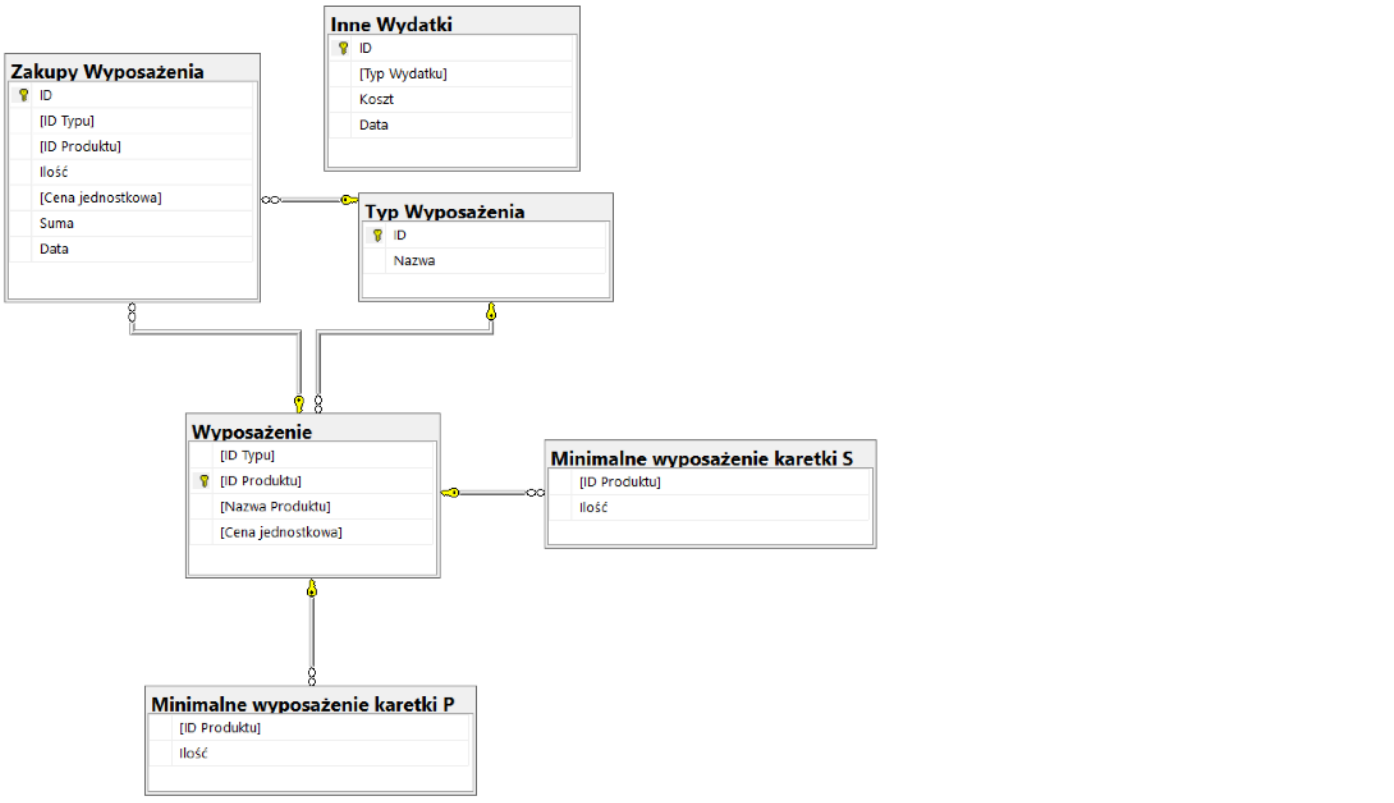
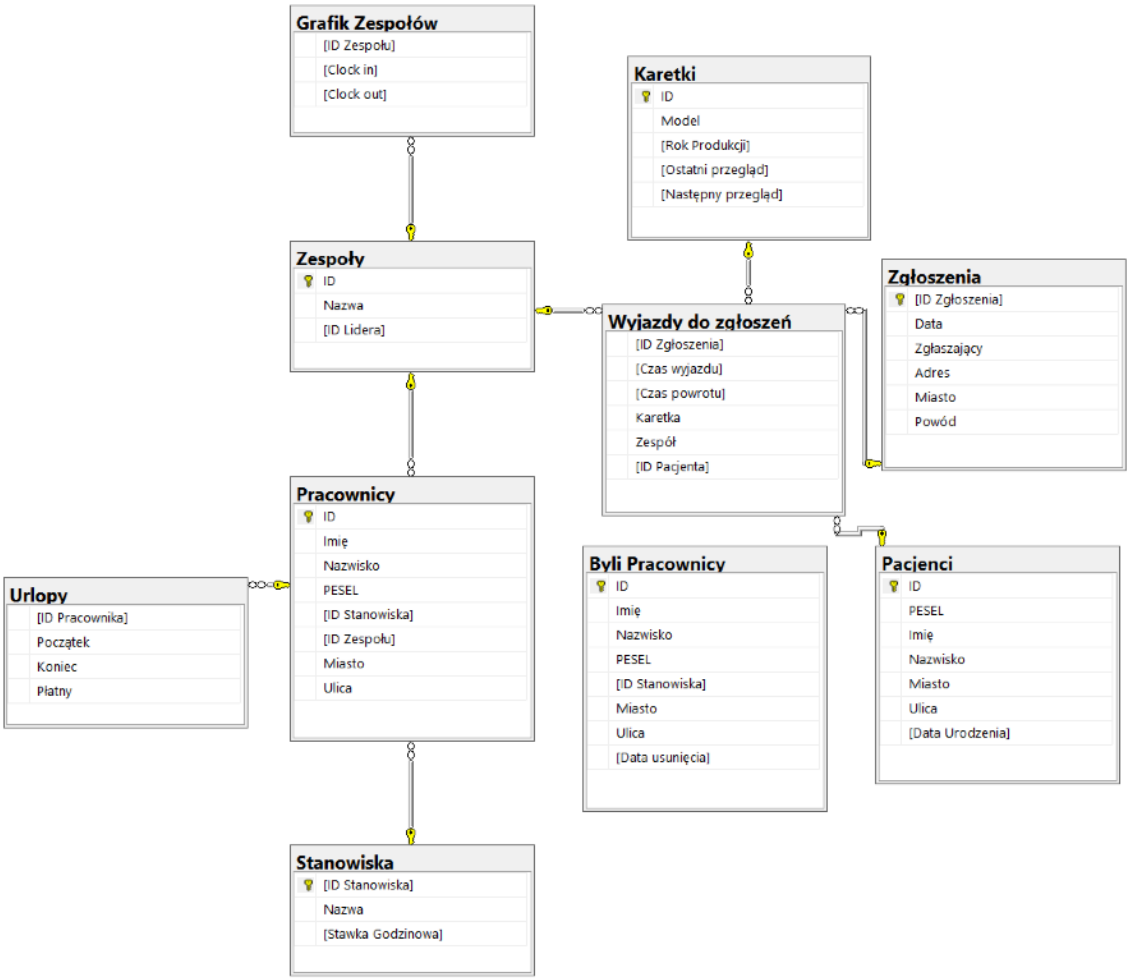
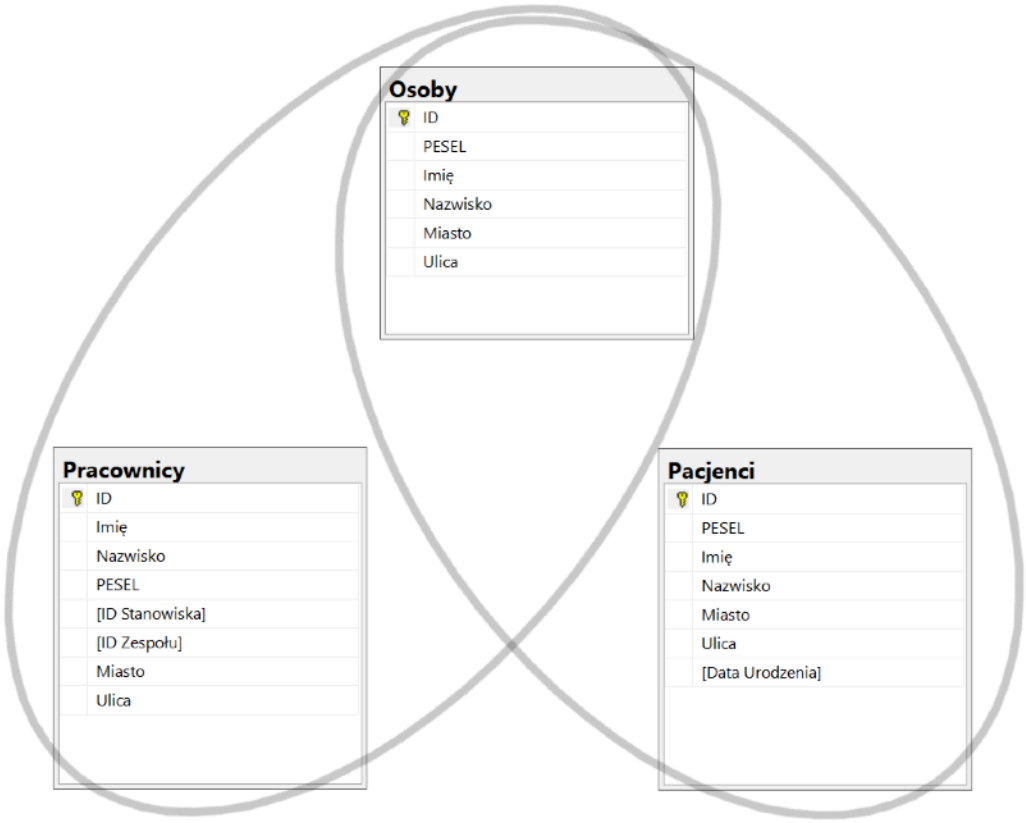

Stawka Godzinowa
Nazwa
ID Stanowiska
Stanowiska
Określa
ID
Nazwa
ID Lidera
Zespoły
Należy
Clock out
Clock in
ID Zespołu
Grafik Zespołów
Rozporządza
Id Zgłoszenia
Data
Powód
Zgłoszenia
Zgłaszający
Adres
Miasto
ID
PESEL
Ulica
Pacjenci
Imię
Nazwisko
Miasto
Data Urodzenia
Dotyczy
ID
Imię
Nazwisko
PESEL
Pracownicy
ID Zespołu
ID Stanowiska
Ulica
Miasto
ID
Imię
Data Urodzenia
Byli Pracownicy
Nazwisko
PESEL
Ulica
ID Stanowiska
ID Zgłoszenia
Zespół
ID Pacjenta
Czas Dojazdu
Wyjazdy Karetki
Czas Wyjazdu
Czas Powrotu
Karetka
Następny Przegląd
ID
Karetki
Model
Rok Produkcji
Ostatni Przegląd
Data
Inne Wydatki
ID
Typ Wydatku
Koszt
Zarządza
ID Pracownika
Początek
Koniec
Płatny
Urlopy
Typ
Nazwa
Typ Wyposażenia
Określa
ID Typu
ID Produktu
Cena Jednostkowa
Nazwa Produktu
Wyposażenie
Dotyczy
ID
Ilość
Suma
Zakupy Wyposażenia
ID Produktu
Cena Jednostkowa
ID Typu
Data
Ilość
Minimalne Wyposażenie Karetki S
ID Produktu
Zawiera się
Ilość
Minimalne Wyposażenie Karetki P
ID Produktu

Diagram Relacji





Schemat dziedziczenia



Tabele:

1.1. Zgłoszenia

Kod tworzący:

```
CREATE TABLE [Zgłoszenia] (
[ID Zgłoszenia] INT PRIMARY KEY,
[Data] DATETIME NOT NULL,
[Zgłaszający] VARCHAR(50),
[Adres] VARCHAR(50),
[Miasto] VARCHAR(50),
[Powód] VARCHAR(128)
);
```

Opis: Tabela zawiera informacje o zgłoszeniach. Każde zgłoszenie musi mieć unikalne ID. Obowiązkowe jest uzupełnienie daty (i odpowiednio ID) zgłoszenia. Pozostale rubryki sa uzupełniane na podstawie otrzymanych informacji. Brak informacji zaznaczany jest nullem.

1.2 Wyjazdy do zgłoszeń

Kod tworzący:

```
CREATE TABLE [Wyjazdy do zgłoszeń] (
[ID Zgłoszenia] INT NOT NULL,
[Czas wyjazdu] DATETIME NOT NULL,
[Czas powrotu] DATETIME NOT NULL,
[Karetka] INT NOT NULL,
[Zespół] INT NOT NULL,
[ID Pacjenta] INT,
   CONSTRAINT FK_IDzgłoszenia FOREIGN KEY ([ID Zgłoszenia])
   REFERENCES Zgłoszenia([ID Zgłoszenia]),
   CONSTRAINT FK_karetka FOREIGN KEY ([Karetka])
   REFERENCES Karetki([ID]),
   CONSTRAINT FK_IDzespołu2 FOREIGN KEY ([Zespół])
   REFERENCES Zespoły([ID]),
```

```
    CONSTRAINT FK_IDpacjenta FOREIGN KEY ([ID Pacjenta])
   REFERENCES Pacjenci([ID])
);
```

Opis: Tabela zawiera informacje o odbytych wyjazdach do przyjętych zgłoszeń, które bezposrednio połączone są z ID Zgłoszenia. Tabela informuje także o czasie wyjazdu, powrotu, która karetka i ktory zespol zostaly wyslane oraz ID Pacjenta. NULL w miejscu ID Pacjenta oznacza, ze pacjent nie zostal przyjety do szpitala- a co sie z tym wiąże; nie zostaly zapisane jego dane.

1.3 Pacjenci

Kod Tworzacy:

```
CREATE TABLE [Pacjenci](
[ID] INT PRIMARY Key,
[PESEL] VARCHAR(11),
[Imię] VARCHAR(50),
[Nazwisko] VARCHAR(50),
[Miasto] VARCHAR(50),
[Ulica] VARCHAR(50),
[Data Urodzenia] date,
CONSTRAINT checkPesel CHECK (dbo.validPesel(PESEL) = 1)
);
```

Opis: Tabela zawiera informacje o przyjetych pacjentach i wszystkich otrzymanych danych ich dotyczacych. Poprawność numeru pesel jest sprawdzana specjalną funkcją.

2.1 Karetki

Kod Tworzacy:

```
CREATE TABLE [Karetki] (
[ID] INT PRIMARY KEY,
[Model] VARCHAR(3) NOT NULL,
[Rok Produkcji] date NOT NULL,
[Ostatni przegląd] date NOT NULL,
[Następny przegląd] date NOT NULL
);
```

Opis: Tabela zawiera model pojazdu oraz jego rok produkcji, datę ostatniego przeglądu oraz do kiedy powinniśmy odbyć kolejny przegląd - podczas wstawiania i edytowania datę wylicza wyzwalacz [nastPrzegląd].

2.2 Typ Wyposażenia

Kod tworzacy:

```
CREATE TABLE [Typ Wyposażenia](
[ID] INT PRIMARY Key,
[Nazwa] VARCHAR(50) NOT NULL
);
```

Opis: Zawiera liste wykorzystywanych typów wyposażenia pogotowia oraz karetek.

2.3 Wyposażenie

Kod Tworzący:

```
CREATE TABLE [Wyposażenie] (
[ID Typu] INT,
[ID Produktu] INT PRIMARY KEY,
[Nazwa Produktu] VARCHAR(50) NOT NULL,
[Cena jednostkowa] MONEY NOT NULL,
   CONSTRAINT FK_IDTypwyp FOREIGN KEY ([ID Typu])
   REFERENCES [Typ Wyposażenia] ([ID])
```

```
);
```

Opis: Tabela jest spisem produktów jakie mogą znajdować się w wyposażeniu karetki oraz używa się w pracy ratownika medycznego. Określa też cenę produktu.

2.4 Minimalne Wyposazenie Karetki P

Kod Tworzacy:

```
CREATE TABLE [Minimalne wyposażenie karetki P] (
[ID Produktu] INT NOT NULL,
[Ilość] INT NOT NULL
   CONSTRAINT FK_IDproduktu FOREIGN KEY ([ID Produktu])
   REFERENCES [Wyposażenie] ([ID Produktu])
);
```

Opis: Zawiera informacje o wymaganym minimalnym wyposażeniu karetki typu P (Typ Podstawowy). Celem tabeli jest ułatwienie prawidłowego uzupełniania wyposażenia karetki.

2.5 Minimalne Wyposażenie Karetki S

Kod tworzacy:

```
CREATE TABLE [Minimalne wyposażenie karetki S] (
[ID Produktu] INT NOT NULL,
[Ilość] INT NOT NULL
   CONSTRAINT FK_IDproduktuS FOREIGN KEY ([ID Produktu])
   REFERENCES [Wyposażenie] ([ID Produktu])
);
```

Opis: Zawiera informacje o wymaganym minimalnym wyposażeniu karetki typu S (Typ Specjalistyczny). Celem tabeli jest ułatwienie prawidłowego uzupełniania wyposażenia karetki.

2.6 Zakupy Wyposazenia

Kod tworzacy:

```
CREATE TABLE [Zakupy Wyposażenia] (
[ID] INT PRIMARY KEY,
[ID Typu] INT,
[ID Produktu] INT,
[Ilość] INT NOT NULL,
[Cena jednostkowa] MONEY NOT NULL,
[Suma] MONEY NOT NULL,
[Data] DATE NOT NULL
   CONSTRAINT FK_IDtypzakupy FOREIGN KEY ([ID Typu])
   REFERENCES [Typ Wyposażenia] ([ID]),
   CONSTRAINT FK_IDprduktuZakup FOREIGN KEY ([ID Produktu])
   REFERENCES [Wyposażenie] ([ID Produktu])
);
```

Opis: Tabela informuje o zrobionych zakupach wyposażenia danego dnia. Zawiera m.in. informacje o ilości zakupionego produktu, cenie po jakiej został on kupiony oraz jaka wyszła nam tego suma.

2.7 Inne Wydatki

Kod tworzacy:

```
CREATE TABLE [Inne Wydatki] (
 [ID] INT PRIMARY KEY,
 [Typ Wydatku] VARCHAR(50) NOT NULL,
 [Koszt] MONEY NOT NULL,
 [Data] DATE NOT NULL
);
```

Opis: Tabela zawiera informacje o wydatkach, ktore nie mieszcza się w kryteriach wyposażenia. Określa datę i kwotę zakupu, oraz krótki, słowny opis stanowiący typ.

3.1 Pracownicy

Kod Tworzący:

```
CREATE TABLE [Pracownicy] (
[ID] INT PRIMARY KEY,
[Imię] VARCHAR(50) NOT NULL,
[Nazwisko] VARCHAR(50) NOT NULL,
[PESEL] VARCHAR(11) NOT NULL,
[ID Stanowiska] INT NOT NULL,
[ID Zespołu] INT,
[Miasto] VARCHAR(50) NOT NULL,
[Ulica] VARCHAR(50) NOT NULL,
   CONSTRAINT FK_IDstanowiska FOREIGN KEY ([ID Stanowiska])
   REFERENCES Stanowiska([ID Stanowiska]),
   CONSTRAINT FK_IDzespołu FOREIGN KEY ([ID Zespołu])
   REFERENCES Zespoły([ID]),
   CONSTRAINT checkPeselPracownicy CHECK
(dbo.validPesel(PESEL) = 1)
);
```

Opis: Tabela zawiera spis aktualnych pracowników oraz ich osobiste informacje.

3.2 Stanowiska

Kod tworzący:

```
CREATE TABLE [Stanowiska] (
 [ID Stanowiska] INT PRIMARY KEY,
 [Nazwa] VARCHAR(60) NOT NULL,
 [Stawka Godzinowa] MONEY NOT NULL
);
```

Opis: Tabela zawiera spis dostepnych stanowisk pracowniczych, oraz jaka jest ich aktualna stawka godzinowa.

3.3 Zespoły

Kod tworzący:

```
CREATE TABLE [Zespoły] (
[ID] INT PRIMARY KEY,
[Nazwa] VARCHAR(50) NOT NULL,
[ID Lidera] INT NOT NULL
CONSTRAINT FK_IDLidera FOREIGN KEY ([ID Lidera])
REFERENCES [Pracownicy] ([ID])
);
```

Opis: Tabela zawiera klucz spis zespołów. Informuje nas o nazwie zespołu oraz kto jest liderem konkretnej drużyny.

3.4 Grafik Zespołów

Kod tworzący:

```
CREATE TABLE [Grafik Zespołów] (
[ID Zespołu] INT NOT NULL,
[Clock in] DATETIME NOT NULL,
[Clock out] DATETIME NOT NULL
  CONSTRAINT FK_IDzespołuGrafik FOREIGN KEY ([ID Zespołu])
   REFERENCES Zespoły([ID])
);
```

Opis: Tabela zawiera informacje o zaplanowanych i minionych zmianach zespołów. Każda zmiana jest określona datą i godziną rozpoczęcia oraz zakończenia.

3.5 Urlopy

Kod tworzący:

```
CREATE TABLE [Urlopy] (
[ID Pracownika] INT NOT NULL,
```

```
[Początek] date NOT NULL,
[Koniec] date NOT NULL,
[Płatny] BIT NOT NULL
   CONSTRAINT FK_IDpracownika FOREIGN KEY ([ID Pracownika])
   REFERENCES Pracownicy([ID]),
   CONSTRAINT checkOverlapVacation CHECK
(dbo.check_overlap_vacation() = 0)
);
```

Opis: Tabela zawiera informacje o minionych i zaplanowanych urlopach pracowników. Dwóch pracowników z jednego zespołu nie może mieć nakładajacych się na siebie urlopów i jest to sprawdzane przy pomocy funkcji *dbo.check_overlap_vacation*.

3.6 Byli Pracownicy

Kod tworzący:

```
CREATE TABLE [Byli Pracownicy] (
[ID] INT PRIMARY KEY,
[Imię] VARCHAR(50),
[Nazwisko] VARCHAR(50),
[PESEL] VARCHAR(11),
[ID Stanowiska] INT NOT NULL,
[Miasto] VARCHAR(50),
[Ulica] VARCHAR(50),
[Data usunięcia] DATE
);
```

Opis: Tabela zawiera informacje o byłych pracownikach pogotowia.

Widoki:

Koszty Dzienne

Kod tworzacy:

```
CREATE VIEW [Koszty Dzienne] AS
SELECT IIF(SumaWypos.DATA IS NULL, SumaInne.DATA,
SumaWypos.DATA) AS Data, IIF(SumaWypos.[Suma Dzienna] IS
NULL, SumaInne.[Suma Dzienna], IIF(SumaInne.[Suma Dzienna]
IS NULL,SumaWypos.[Suma Dzienna], SumaWypos.[Suma Dzienna]+
SumaInne.[Suma Dzienna])) AS [Suma Dzienna] FROM
(
   SELECT DATA AS DATA, SUM([SUMA]) [Suma Dzienna] FROM
[Zakupy Wyposażenia]
   GROUP BY (DATA)
) AS SumaWypos
FULL OUTER JOIN
(
   SELECT DATA, SUM(Koszt) [Suma Dzienna] FROM [Inne
Wydatki]
   GROUP BY (DATA)
) AS SumaInne
ON SumaInne.Data = SumaWypos.Data
```

Opis: Widok sumuje wydatki z tabel [Zakupy Wyposażenia] oraz [Inne Wydatki] i grupuje je dniami. Pozwala nam to na szybki dostęp do poniesionych danego dnia kosztów, oraz analizy jego na tle reszty dni.

Średnia i Max kosztów dziennych

Kod tworzący:

```
CREATE VIEW [ŚrednieMax koszty dzienne] AS
SELECT AVG([Suma Dzienna]) [Średnia Wydatków], MAX([Suma
Dzienna]) AS [Max Dzienny]  FROM [Koszty Dzienne]
```

Opis: Widok wypisuje średnie i maksymalne wydatki dzienne. Tak jak poprzedni widok pozwala nam na analizę wydatków pogotowia.

Statystyki Zespołów

Kod tworzący:

```
CREATE VIEW [Statystyki Zespołów] AS
   SELECT [Zespół], COUNT([Zespół]) AS [Ilość Wyjazdów] FROM
[Wyjazdy do zgłoszeń]
   GROUP BY [Zespół]
```

Opis: Widok wypisuje ile wyjazdow łącznie odbył każdy zespół. Pozwala określić który zespół był najaktywniejszy.

Skład Zespołów

Kod tworzacy:

```
CREATE VIEW [Skład Zespołów] AS
SELECT Z.Nazwa AS [Nazwa Zespołu], P.Imię AS [Imię],
P.Nazwisko [Nazwisko], S.Nazwa [Stanowisko]
FROM Zespoły Z
   LEFT JOIN
   Pracownicy P ON (Z.ID = P.[ID Zespołu])
   JOIN
   Stanowiska S ON (P.[ID Stanowiska] = S.[ID Stanowiska])
```

Opis: Widok wyświetla wszystkich ratowników i do jakich zespołów należą.

Wyjazdy dzienne Statystyki

Kod tworzacy:

```
CREATE VIEW [Wyjazdy Dzienne] AS
SELECT LiczbaWyj.Dzień AS DATA, IIF(LiczbaP.LiczbaP IS NULL,
0, LiczbaP.LiczbaP) AS [Karetki typu P], IIF(LiczbaS.LiczbaS
IS NULL, 0, LiczbaS.LiczbaS) AS [Karetki typu S] FROM
```

```
(SELECT CONVERT(DATE, [Czas wyjazdu]) Dzień, COUNT(*)
LiczbaWyjazdow FROM [Wyjazdy do zgłoszeń]
GROUP BY CONVERT(DATE, [Czas wyjazdu])) AS LiczbaWyj
LEFT JOIN
(SELECT CONVERT(DATE, W.[Czas wyjazdu]) Dzień, COUNT(*) AS
LiczbaP FROM [Wyjazdy do zgłoszeń] W, Karetki K
WHERE (W.Karetka = K.ID AND K.Model = 'P')
GROUP BY CONVERT(DATE, W.[Czas wyjazdu])) AS LiczbaP
ON LiczbaP.Dzień = LiczbaWyj.Dzień
LEFT JOIN
(SELECT CONVERT(DATE, W.[Czas wyjazdu]) Dzień, COUNT(*) AS
LiczbaS FROM [Wyjazdy do zgłoszeń] W, Karetki K
WHERE (W.Karetka = K.ID AND K.Model = 'S')
GROUP BY CONVERT(DATE, W.[Czas wyjazdu]) ) AS LiczbaS
ON LiczbaWyj.Dzień = LiczbaS.Dzień
```

Opis: Widok wypisuje jakiego dnia ile wyjazdów odbyły karetki każdego typu.

Zgłoszenia bez wyjazdu

Kod tworzacy:

```
CREATE VIEW [Zgłoszenia bez wyjazdu] AS
SELECT [Zgłoszenia bez wyjazdu].[ID Zgłoszenia], Z.[Powód]
FROM
   (SELECT [ID Zgłoszenia] FROM Zgłoszenia WHERE [ID
Zgłoszenia] NOT IN
   ( SELECT [ID Zgłoszenia] FROM [Wyjazdy do zgłoszeń] )) AS
[Zgłoszenia bez wyjazdu]
   LEFT JOIN
      Zgłoszenia Z ON (Z.[ID Zgłoszenia]=[Zgłoszenia bez
wyjazdu].[ID Zgłoszenia])
```

Opis: Wypisuje ID zgłoszeń, do których nie nastąpił wyjazd karetki oraz jaki został podany powód podczas zgłoszenia. Pozwala sprawdzić czy nastąpiły słuszne decyzje o braku wyjazdu.

Zgloszenia a wyjazdy dzienne

Kod tworzacy:

```
CREATE VIEW [Zgłoszenia a wyjazdy dzienne] AS
   SELECT
   (SELECT COUNT(*) FROM Zgłoszenia) AS [Ilość Zgłoszeń],
   (SELECT COUNT(*) FROM [Wyjazdy do zgłoszeń]) AS [Ilość
Wyjazdów],
   (SELECT CAST((
       CAST((SELECT COUNT(*) FROM [Wyjazdy do zgłoszeń]) AS
FLOAT) / CAST((SELECT COUNT(*) FROM Zgłoszenia) AS FLOAT)
      *100) AS DECIMAL(10,2) )) AS [Procent]
```

Opis: Widok określa ile nastąpiło zgłoszeń danego dnia oraz wyjazdów, i jaki jest ich stosunek wyrażony w procentach.

Funkcje

check_birthdate

Kod tworzący:

```
CREATE FUNCTION check_birthdate
(
   @data DATE
)
RETURNS BIT
AS
BEGIN
   DECLARE @result BIT;
   SET @result = 1;

   IF (@data > GETDATE())
   BEGIN
       SET @result = 0;
   END
   IF (@result = 0)
   BEGIN
       return cast('Została wpisana zła data' as bit);
   END
   RETURN @result;
END
```

Opis: Funkcja sprawdza czy do tabeli została wpisana poprawna data urodzenia. W przypadku złej daty wykorzystuje błąd castowania w celu wypisania informacji przydatnej w debugowaniu.

check_overlap_vacation

Kod tworzący:

```
CREATE FUNCTION check_overlap_vacation()
RETURNS BIT
AS
```

```
BEGIN
   DECLARE @overlap BIT;
   SET @overlap = 0;

   SELECT @overlap = COUNT(*)
   FROM Urlopy
  WHERE EXISTS (SELECT 1 FROM Urlopy B, Urlopy A, Pracownicy
Pa, Pracownicy Pb WHERE A.[ID Pracownika] <> B.[ID
Pracownika] AND A.[ID Pracownika] = Pa.ID AND B.[ID
Pracownika] = Pb.ID AND Pb.[ID Zespołu]=Pa.[ID Zespołu] AND
((A.[Początek] <= B.[Koniec] AND A.[Początek] >= B.
[Początek]) OR (A.[Koniec] <= B.[Koniec] AND A.[Koniec] >=
B.[Początek])))

   IF @overlap > 0
       SET @overlap = 1

   RETURN @overlap;
END
```

Opis: Funkcja spradza czy dwóch pracowników z jednego zespołu nie ma zaplanowanych wakacji w tym samym czasie - co jest sytuacją niedopuszczalną, ponieważ zespół miałby zbyt mało członków w danym czasie.

validPesel

Kod tworzący:

```
CREATE FUNCTION validPesel
(
   @pesel VARCHAR(11)
)
RETURNS BIT
AS
begin
```

```
    IF ISNUMERIC(@pesel) = 0
        RETURN 0

    DECLARE @wagi AS TABLE (
        Pozycja TINYINT IDENTITY(1,1) NOT NULL,
        Waga TINYINT NOT NULL
    )

    INSERT INTO @wagi VALUES
    (1), (3), (7), (9), (1), (3), (7), (9), (1), (3), (1)

    IF (
        SELECT SUM(CONVERT(TINYINT, SUBSTRING(@pesel,
Pozycja, 1)) * Waga) % 10 FROM @wagi
    ) = 0
        RETURN 1;

    RETURN 0;
end
```

Opis: Funkcja sprawdza poprawność numeru PESEL.

Procedury składowane

Informacje o wyjeździe karetki

Kod tworzący:

```
CREATE PROCEDURE [Dojazd]
    @ID INT
AS
begin
    SELECT WK.[ID zgłoszenia], P.Imię + ' ' + P.Nazwisko AS
[Kierowca], Z.Data [Czas zgłoszenia], WK.[Czas wyjazdu], WK.[Czas
dojazdu], DATEDIFF(MINUTE,WK.[Czas wyjazdu], WK.[Czas dojazdu])
AS [Długość dojazu minuty], DATEDIFF(SECOND,WK.[Czas wyjazdu],
WK.[Czas dojazdu]) AS [Długość dojazu sekundy] ,Z.Adres, Z.Miasto
FROM Zgłoszenia Z
        JOIN
    [Wyjazdy do zgłoszeń] WK ON (WK.[ID Zgłoszenia] = Z.[ID
Zgłoszenia])
        JOIN
    [Pracownicy] P ON (P.[ID Zespołu] = WK.[Zespół])
   WHERE Z.[ID Zgłoszenia] = @ID AND P.[ID Stanowiska] = 5
   GROUP BY WK.[ID Zgłoszenia], P.Imię, P.Nazwisko, Z.[Data], WK.
[Czas wyjazdu], WK.[Czas dojazdu], Z.Adres, Z.Miasto
end
```

Opis: Procedura daje nam informacje na podstawie ID konkretnego zgłoszenia o wykonanym wyjeździe karetki. Wyświetla m.in. informacje o kierowcy oraz liczy ile zajął dojazd w sekundach i minutach na konkretny adres.

Wypłata za okres dla Pracownika

Kod tworzący:

```
     CREATE PROCEDURE [Wypłata]
```

```
    @From DATE,
    @To DATE,
    @EmployeeID INT
AS
begin
    DECLARE @temp_table TABLE (
        ID int,
        Imię varchar(50)   ,
        Nazwisko VARCHAR(50),
        [Przepracowane Godziny] INT
    );
    INSERT INTO @temp_table
    EXEC [Liczba Przepracowanych Godzin] @From, @To,
@EmployeeID

    SELECT TT.ID, TT.Imię, TT.Nazwisko, TT.[Przepracowane
Godziny], S.[Stawka Godzinowa], TT.[Przepracowane
godziny]*S.[Stawka Godzinowa] AS [Wypłata] FROM @temp_table
TT
        JOIN
    Pracownicy P ON (P.ID = TT.ID)
        JOIN
    Stanowiska S ON (S.[ID Stanowiska] = P.[ID Stanowiska])

    WHERE @EmployeeID = P.ID
end
```

Opis: Procedura pobiera argumenty zawierające początek oraz koniec okresu dla jakiego chcemy wyliczyc wypłatę oraz ID dla jakiego pracownika. Wyświetla także liczbę przepracowanych godzin pobranych z procedury *[Liczba Przepracowanych Godzin]* i stawkę godzinową pracownika.

Kiedy ostatni raz został zakupiony ten przedmiot i w jakiej cenie

Kod tworzący:

```
CREATE PROCEDURE [Ostatni zakup]
   @Name VARCHAR(50)
   AS
begin
   SELECT TOP 1 W.[Nazwa Produktu], ZW.[Data], ZW.[Cena
jednostkowa] FROM Wyposażenie W
   JOIN [Zakupy Wyposażenia] ZW ON (ZW.[ID Produktu] = W.[ID
Produktu])
   WHERE W.[Nazwa Produktu] = @Name
   ORDER BY ZW.[Data] DESC
end
```

Opis: Dla danego argumentu stanowiącego ID Produktu procedura wyświetla kiedy ostatni raz kupiony został dany produkt i w jakiej cenie.

Ile osób jest w danym zespole

Kod tworzący:

```
CREATE PROCEDURE [Liczebność Zespołów]
   @TeamID INT
AS
begin
   SELECT Z.ID, Z.Nazwa, COUNT(DISTINCT P.ID) AS [Ilość
członków] FROM Zespoły Z
       JOIN
   Pracownicy P ON (Z.ID = P.[ID Zespołu])
  WHERE Z.ID = @TeamID
  GROUP BY Z.Nazwa, Z.ID
end
```

Opis: Dla danego argumentu stanowiącego ID Zespołu procedura wyświetla nam liczebność danego zespołu oraz jego nazwę.

Ile godzin przepracował pracownik w danym okresie

Kod tworzący:

```
ALTER PROCEDURE [Liczba Przepracowanych Godzin]
   @From DATE,
   @To DATE,
   @EmployeeID INT
AS
begin
   SELECT P.ID, P.Imię, P.Nazwisko, SUM(DATEDIFF(HOUR,GZ.
[Clock in],GZ.[Clock out])) AS [Przepracowane godziny]
   FROM [Pracownicy] P
       JOIN
   Zespoły Z ON (P.[ID Zespołu] = Z.[ID])
       JOIN
   [Grafik Zespołów] GZ ON (GZ.[ID Zespołu] = Z.[ID])
   WHERE @EmployeeID = P.[ID] AND @From < GZ.[Clock in] AND
@To > GZ.[Clock out]
   GROUP BY P.ID, P.Imię, P.Nazwisko
end
```

Opis: Procedura pobiera argumenty zawierające początek oraz koniec okresu dla jakiego chcemy wyliczyc ile godzin przepracował dany pracownik. Wyświela m.in. ilość godzin, zakres oraz imię i nazwisko pracownika.

Grafik zespołów na konkretny dzień

Kod tworzący:

```
CREATE PROCEDURE [Grafik Dzień]
   @Date DATE
   AS
begin
   SELECT Z.ID, Z.Nazwa FROM Zespoły Z
   JOIN [Grafik Zespołów] GZ ON (GZ.[ID Zespołu] = Z.ID)
   WHERE CAST(GZ.[Clock in] as date) = @Date
end
```

Opis: Procedura datę dla jakiej chcemy otrzymać informację o grafiku. Wyświetla nazwy oraz ID Zespołów, które pracują w dany dzień.

Wyzwalacze

Suma w Zakupy wyposażenia

Kod tworzący:

```
CREATE TRIGGER sumazakup
ON [Zakupy Wyposażenia]
AFTER INSERT, UPDATE
AS
BEGIN
   UPDATE [Zakupy Wyposażenia]
   SET SUMA = i.[Ilość]*i.[Cena jednostkowa]
   FROM [Zakupy Wyposażenia]
   INNER JOIN inserted i on i.ID = [Zakupy Wyposażenia].ID
END
GO
```

Opis: Automatycznie oblicza sumę na podstawie ilości i ceny jednostkowej po insercie i updacie w tabeli Zakupy Wyposazenia.

Najpóźniejsza data następnego przeglądu karetki

Kod tworzący:

```
CREATE TRIGGER nastPrzegląd
ON [Karetki]
AFTER UPDATE, INSERT
AS
BEGIN
   UPDATE [Karetki]
   SET [Następny przegląd] = DATEADD(month, 4, i.[Ostatni
przegląd])
   FROM [Karetki]
   INNER JOIN inserted i on i.ID = [Karetki].ID
END
GO
```

Opis: Automatycznie oblicza datę nastepnego przeglądu na podstawie ostatniego przeglądu w tabeli Karetki.

Sprawdzanie czy wpisana stawka jest nieujemna

Kod tworzący:

```
CREATE TRIGGER NegativeValue_UPDATE ON Stanowiska
   AFTER UPDATE, INSERT
AS
begin
SET NOCOUNT ON;

   UPDATE Stanowiska SET [Stawka Godzinowa] = CASE
WHEN([Stawka Godzinowa] < 0)
       THEN ABS([Stawka Godzinowa])
       ELSE [Stawka Godzinowa]
   END
end
```

Opis: Sprawdza czy stawka w tabeli Stanowiska jest dodatnia. Gdy jest ujemna, poprawia znak.

Automatyczny zapis daty w tabeli Zgłoszenia

Kod tworzący:

```
CREATE TRIGGER CzasZgłoszenia
ON Zgłoszenia
AFTER INSERT
AS
begin
   UPDATE [Zgłoszenia]
   SET [Data] = GETDATE()
   FROM Zgłoszenia Z
       JOIN
   inserted i ON (i.[ID Zgłoszenia] = Z.[ID Zgłoszenia])
```

```
    WHERE Z.[ID Zgłoszenia] = i.[ID Zgłoszenia]
end
GO
```

Opis: Automatycznie ustawia datę podczas wpisywania nowych zgloszeń.

Byli pracownicy

Kod tworzący:

```
CREATE TRIGGER tr_Pracownicy_delete
ON Pracownicy
AFTER DELETE
AS
BEGIN
   SET NOCOUNT ON;
   INSERT INTO [Byli Pracownicy] (ID, Imię, Nazwisko, PESEL,
[ID Stanowiska], Miasto, Ulica, [Data usunięcia])
   SELECT ID, Imię, Nazwisko, PESEL, [ID Stanowiska],
Miasto, Ulica, GETDATE()
   FROM deleted;
END
```

Opis: Automatycznie uzupełnia tabele byli pracownicy o usuniętych pracowników z tabeli Pracownicy.

Typowe zapytania

- ```
  SELECT COUNT([ID]) [Ilość osób na stanowisku] FROM
  Pracownicy
  WHERE [ID Stanowiska] = 3
  ```

- ```
  SELECT TOP 1 [Ostatni przegląd] FROM Karetki
  WHERE ID = 3
  ORDER BY [Ostatni przegląd] DESC
  ```

- ```
  SELECT * FROM Pacjenci
  WHERE Pacjenci.PESEL = '99041635492'
  ```
  ```