

# Spis treści

<b>1. Projekt konceptualny .....</b>	<b>2</b>
1.1. Sformułowanie zadania projektowego .....	2
1.2. Analiza stanu wyjściowego.....	2
1.3. Analiza wymagań użytkownika .....	2
1.4. Określenie scenariuszy użycia, diagram UML .....	4
1.5. Identyfikacja funkcji.....	5
1.6. Propozycja encji i ich atrybutów (Diagram ERD) .....	6
<b>2. Projekt logiczny .....</b>	<b>7</b>
2.1. Przejście z modelu ERD na model relacyjny (projekt logiczny) .....	7
2.2. Normalizacja .....	7
2.3. Diagram relacyjnej bazy danych po normalizacji .....	14
<b>3. Projekt implementacyjny .....</b>	<b>15</b>
3.1. Kod SQL .....	16
3.2. Kwerendy .....	28
3.3. Algebra relacyjna.....	37
<b>4. Określenie kierunków rozwoju aplikacji .....</b>	<b>38</b>
4.1. Literatura:.....	38
<b>5. Podsumowanie .....</b>	<b>39</b>

# **1. Projekt konceptualny**

## **1.1. Sformułowanie zadania projektowego**

Celem projektu jest stworzenie kompleksowej bazy danych dla gabinetu weterynaryjnego, mającej na celu usprawnienie codziennej pracy. System obejmuje rejestrację zwierząt, planowanie wizyt, przydzielanie odpowiedniego lekarza i gabinetu, prowadzenie pełnej dokumentacji medycznej oraz usprawnianie komunikacji z właścicielem zwierząt. Projekt dąży do stworzenia systemu, który nie tylko ułatwi codzienną pracę w gabinecie weterynaryjnym, ale również zwiększy bezpieczeństwo danych, usprawni zarządzanie zasobami oraz uprości codzienne procesy administracyjne.

## **1.2. Analiza stanu wyjściowego**

Tematem projektu jest praca w gabinecie weterynaryjnym, która stanowi centralną platformę wspomagającą codzienne funkcjonowanie lekarzy weterynarii oraz obsługę właścicieli zwierząt. System ten jest zaprojektowany w celu efektywnego zarządzania informacjami dotyczącymi pacjentów oraz ułatwiania procesów związanych z opieką zdrowotną zwierząt. W ramach platformy weterynaryjnej gromadzone są szczegółowe informacje dotyczące każdego pacjenta i jego właściciela. Zapisywane są dane medyczne, historie szczepień, przepisane leki oraz terminy wizyt. Dzięki temu lekarze mają łatwy dostęp do pełnej historii zdrowotnej zwierząt, co umożliwia im skuteczniejszą diagnostykę i leczenie.

## **1.3. Analiza wymagań użytkownika**

### **Właściciel**

- Właściciel ma dostęp do przeglądania dostępnych terminów wizyt oraz historii poprzednich wizyt.
- Właściciel ma dostęp do danych właściciela zwierzęcia oraz możliwość aktualizacji i zarządzania tymi informacjami.
- Właściciel ma możliwość zarejestrowania nowego zwierzęcia, dostarczając niezbędne informacje oraz aktualizowanie i zarządzanie nimi.
- Właściciel ma możliwość zalogowania się, umówienia wizyty i dokonanie płatności.
- Właściciel ma możliwość odwołania umówionej wcześniej wizyty.

### **Lekarz**

- Lekarz ma dostęp do danych zwierzęcia oraz możliwość aktualizacji i zarządzania tymi informacjami.
- Lekarz ma możliwość zarządzania wizytami, dostarczając niezbędne informacje oraz aktualizowanie i zarządzanie nimi.
- Lekarz ma podgląd do płatności oraz umówionych wizyt.
- Lekarz ma możliwość przeglądania zapasów leków oraz uzupełniania braków tych leków.

### **Lekarz – dyrektor**

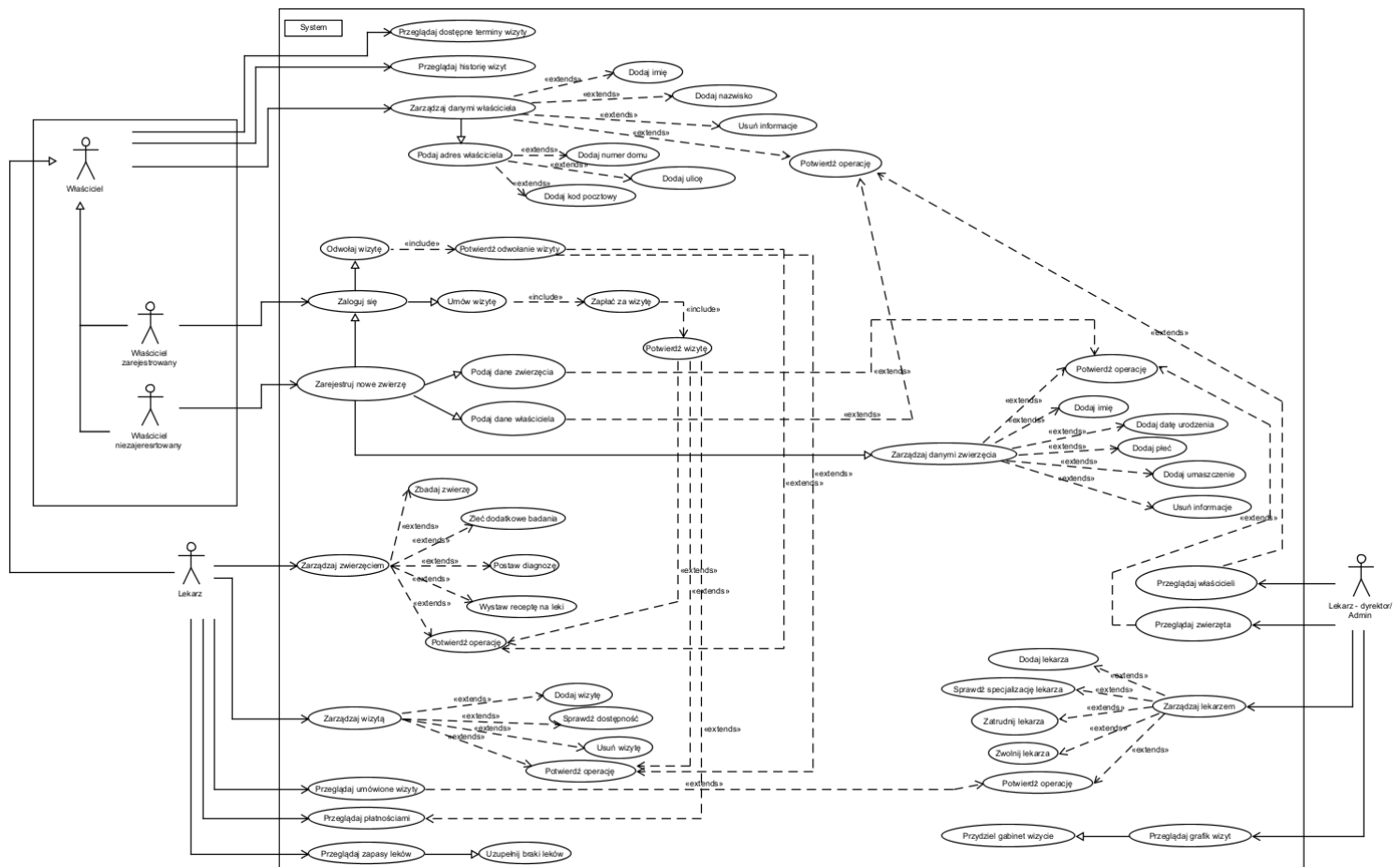
- Admin ma możliwość zarządzania lekarzami, właścicielami zwierząt oraz zwierzętami.

- Admin ma dostęp do danych lekarzy, właścicieli oraz zwierząt. Ma możliwość aktualizacji i zarządzania tymi informacjami.
- Admin ma dostęp do grafiku wizyt oraz możliwość przydzielenia odpowiedniego gabinetu wizycie.

#### 1.4. Określenie scenariuszy użycia, diagram UML

Poniższy diagram UML przedstawia system, który wspomaga pracę w gabinecie weterynaryjnym. Aktorami są: właściciel (zarejestrowany i niezarejestrowany), lekarz i lekarz – dyrektor/Admin, każdy z nich ma przypisane odpowiednie role.

- Lekarz - dyrektor/Admin - dysponuje najwyższymi uprawnieniami, obejmującymi kompleksową kontrolę nad zarządzanym personelem i dostęp do zaawansowanych funkcji systemu.
- Właściciel zarejestrowany – po zalogowaniu na konto, użytkownik zyskuje dodatkowe uprawnienia i możliwości.
- Właściciel niezarejestrowany - ma ograniczony dostęp, obejmujący jedynie funkcje podstawowe.
- Lekarz – posiada szeroki zakres uprawnień w systemie.

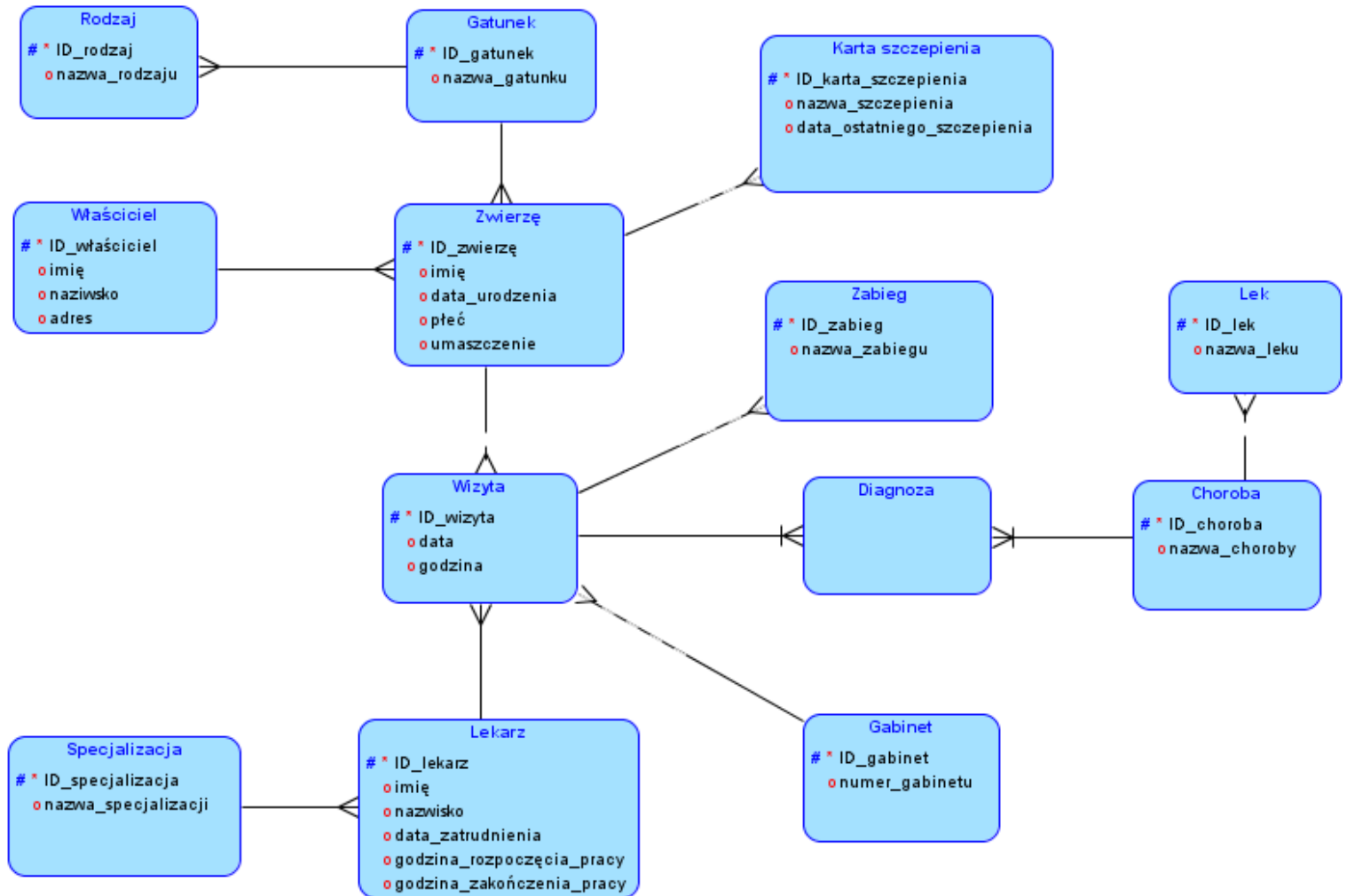


## 1.5. Identyfikacja funkcji

- Przeglądanie dostępnych terminów wizyt – opcja dla właściciela, który może sprawdzić dostępność wolnych terminów wizyt w gabinecie weterynaryjnym.
- Przeglądanie historii wizyt – Możliwość sprawdzenia przez właściciela historii wcześniej odbytych wizyt jego zwierzęcia.
- Zarządzanie danymi właściciela – funkcja umożliwiająca właścicielowi modyfikację informacji o sobie w systemie.
- Rejestracja – możliwość założenia prywatnego konta w systemie, co umożliwia właścicielowi umówienie swojego pupila na wizytę w gabinecie weterynaryjnym już jako właściciel zarejestrowany.
- Logowanie – dostęp do konta przez właściciela, który przeszedł rejestrację. Pozwala na zarządzanie danymi osobowymi, danymi zwierzęcia, a także umawianie, odwoływanie wizyt oraz dokonywanie płatności.
- Zarządzaj danymi zwierzęcia – możliwość wprowadzania i modyfikowania informacji o danym zwierzęciu przez właściciela.
- Zarządzanie zwierzęciem – dodawanie informacji o zdrowiu zwierzęcia oraz przydzielanie odpowiedniej pomocy
- Zarządzanie wizytą – możliwość sprawdzenia umówionych wizyt, przeniesienia ich lub odwołania przez lekarza.
- Przeglądanie wizyt – opcja zajrzenia do grafiku wizyt.
- Przeglądanie płatności – możliwość sprawdzenia przez lekarza opłaconych wizyt.
- Przeglądanie zapasów leków – sprawdzenie przez lekarza dostępności leków i w razie potrzeby uzupełnienie zapasów.
- Zarządzanie lekarzem – funkcja dostępna przez administratora, który może wprowadzić i zmieniać dane o lekarzu.
- Przeglądanie właścicieli – sprawdzenie przez administratora listy zapisanych korzystających z usług przychodni weterynaryjnej.
- Przeglądanie zwierząt – sprawdzenie przez administratora listy pacjentów, czyli zwierząt leczących się w przychodni weterynaryjnej.
- Przeglądanie grafiku wizyt – możliwość sprawdzenia przez administratora harmonogramu pracy lekarzy, wprowadzenie do nich zmian i przydzielenie odpowiednich gabinetów do wizyt.

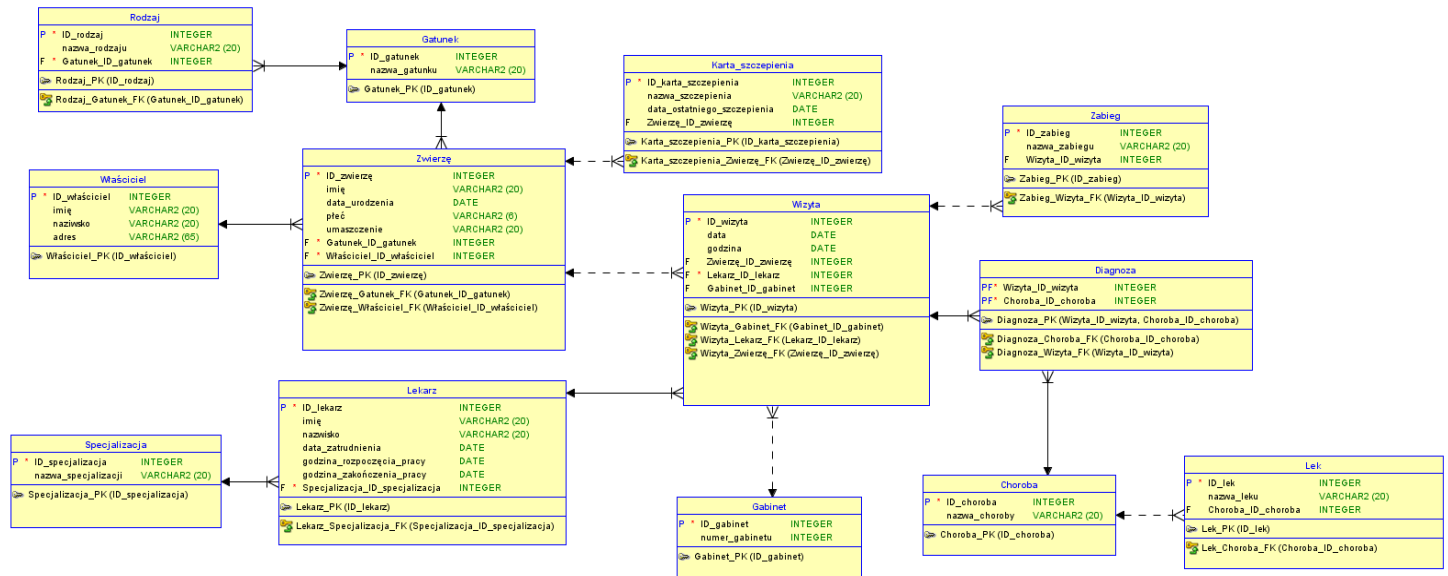
## 1.6. Propozycja encji i ich atrybutów (Diagram ERD)

Zgodnie z przedstawionym poniżej diagramem ERD, stworzono następujące encje: zwierzę, gatunek, rodzaj, właściciel, wizyta, lekarz, specjalizacja, karta szczepień, zabieg, gabinet, choroba, lek. Dodatkowo, ze względu na relacje wiele-do-wielu, wprowadzono tablicę asocjacyjną: diagnoza. Przy doborze atrybutów dla każdej encji oraz analizie relacji między obiektami, zdecydowano się na implementację związków 1:N. Część tych połączeń jest wymagalna, a część opcjonalna, w zależności od konkretnych potrzeb.



## 2. Projekt logiczny

### 2.1. Przejście z modelu ERD na model relacyjny (projekt logiczny)



### 2.2. Normalizacja

- ZWIERZĘ**

ZWIERZĘ
ID_zwierzę
imię
data_urodzenia
pleć
umaszczenie

1NF	Tabela <i>Zwierzę</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Zwierzę</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Zwierzę</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

- **GATUNEK**

<b>GATUNEK</b>
<u>ID_gatunek</u>
nazwa_gatunku

1NF	Tabela <i>Gatunek</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Gatunek</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Gatunek</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

- **RODZAJ**

<b>RODZAJ</b>
<u>ID_rodzaj</u>
nazwa_rodzaju

1NF	Tabela <i>Rodzaj</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Rodzaj</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Rodzaj</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.



- WŁAŚCICIEL

WŁAŚCICIEL
<u>ID_właściciel</u>
imię
nazwisko
adres



WŁAŚCICIEL
<u>ID_właściciel</u>
imię
nazwisko
adres
kod_pocztowy
miasto
ulica
numer_domu
numer_mieszkania

1NF	Początkowo tabela "właściciel" nie była w pierwszej normalnej formie (1NF), ponieważ atrybut "adres" zawierał więcej niż jedną wartość (kolumny kod pocztowy, miasto, ulica, numer domu, numer mieszkania). W celu dostosowania do 1NF, atrybut "adres" został rozdzielony na kilka kolumn, aby każda z nich zawierała pojedynczą wartość. Po tej operacji tabela nazywana była nadal "właściciel", a jej atrybuty to: ID_właściciel, imię, nazwisko, ulica, numer domu i numer mieszkania, kod pocztowy, miasto.
-----	---

WŁAŚCICIEL
<u>ID_właściciel</u>
imię
nazwisko
adres
kod_pocztowy
miasto
ulica
numer_domu
numer_mieszkania



WŁAŚCICIEL
<u>ID_właściciel</u>
imię
nazwisko
ulica
numer_domu
numer_mieszkania



ADRES
<u>ID_adres</u>
kod_pocztowy
miasto

2NF	W celu osiągnięcia drugiej normalnej formy (2NF), tabela "właściciel" została poddana rozdzieleniu na dwie oddzielne tabele, eliminując redundancję danych i wprowadzając unikalne identyfikatory.
3NF	W celu osiągnięcia trzeciej normalnej formy (3NF), atrybuty związane z adresem, tj. "kod pocztowy" i "miasto", zostały przeniesione do oddzielnej tabeli "adres". Wprowadzono unikalny identyfikator "ID_adres" dla tabeli "adres", co pozwoliło na eliminację zależności tranzycyjnych.

- **WIZYTA**

WIZYTA
<u>ID wizyta</u>
data
godzina

1NF	Tabela <i>Wizyta</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Wizyta</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Wizyta</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

- **LEKARZ**

LEKARZ
<u>ID lekarz</u>
imię
nazwisko
data_zatrudnienia
godzina_rozpoczęcia_pracy
godzina_zakończenia_pracy

1NF	Tabela <i>Lekarz</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Lekarz</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Lekarz</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

- **SPECJALIZACJA**

SPECJALIZACJA
---------------

ID_specjalizacja
nazwa_specjalizacji

1NF	Tabela <i>Specjalizacja</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Specjalizacja</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Specjalizacja</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

- KARTA SZCZEPIENIA**

<b>KARTA SZCZEPIENIA</b>
ID_karta_szczepienia
nazwa_szczepienia
data_ostatniego_szczepienia

1NF	Tabela <i>Karta szczepienia</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Karta szczepienia</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Karta szczepienia</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

- **ZABIEG**

<b>ZABIEG</b>
<u>ID_zabieg</u>
nazwa_zabiegu

1NF	Tabela <i>Zabieg</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Zabieg</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Zabieg</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

- **GABINET**

<b>GABINET</b>
<u>ID_gabinet</u>
numer_gabinetu

1NF	Tabela <i>Gatunek</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Gatunek</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Gatunek</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

- **LEK**

LEK
<u>ID_lek</u>
nazwa_leku

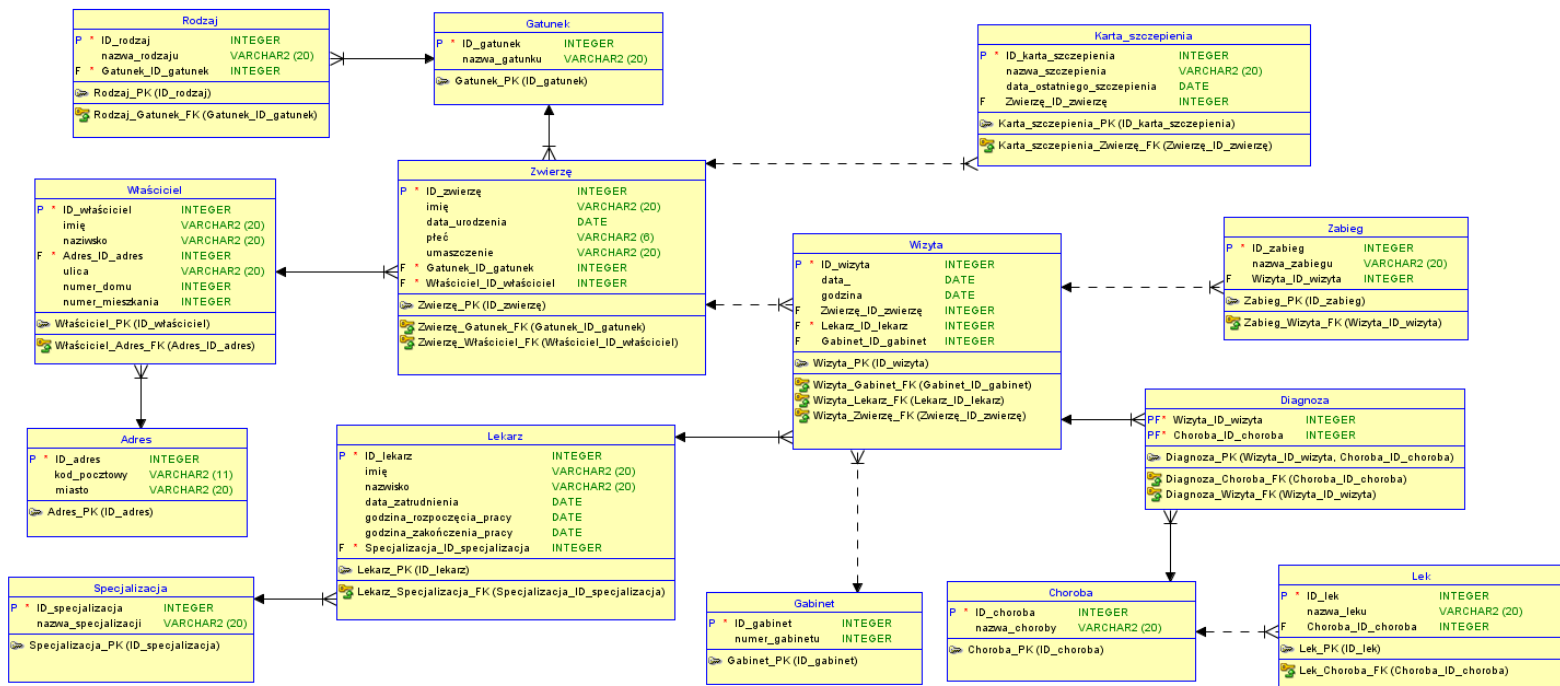
1NF	Tabela <i>Lek</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Lek</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Lek</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

- **CHOROBA**

CHOROBA
<u>ID_choroba</u>
nazwa_choroba

1NF	Tabela <i>Choroba</i> spełnia warunki pierwszej normalnej formy (1NF), ponieważ każda komórka w kolumnach zawiera pojedynczą wartość, co eliminuje wielokrotne wartości w tych kolumnach. Każda kolumna została dostosowana w taki sposób, aby zawierała tylko atomowe wartości.
2NF	Tabela <i>Choroba</i> spełnia warunki drugiej normalnej formy (2NF), ponieważ dane dotyczące różnych jednostek informacyjnych zostały odpowiednio rozdzielone między różne tabele. Wprowadzenie osobnych tabel dla poszczególnych jednostek eliminuje redundancję informacji i zapewnia, że każda tabela w 2NF zawiera unikalny identyfikator oraz dane związane tylko z daną jednostką.
3NF	Tabela <i>Choroba</i> spełnia warunki trzeciej normalnej formy (3NF), gdyż wszystkie pola danych w tabeli są funkcjonalnie zależne tylko od klucza głównego. Proces ten eliminuje wszelkie zależności tranzycyjne, co pozwala uniknąć potencjalnych problemów związanych z redundancją danych.

## 2.3. Diagram relacyjnej bazy danych po normalizacji



### 3. Projekt implementacyjny

Charakteryzacja tabel:

1. **adres:** Informacje o adresach, zawiera id\_adres, kod\_pocztowy, i miasto.
2. **choroba:** Dane na temat różnych chorób, z id\_choroba i nazwa\_choroby.
3. **diagnoza:** Tabela łącznikowa między wizytami lekarskimi a chorobami, zawiera wizyta\_id\_wizyta i choroba\_id\_choroba.
4. **gabinet:** Informacje o gabinetach, z id\_gabinet i numer\_gabinetu.
5. **gatunek:** Dane na temat gatunków zwierząt, z id\_gatunek i nazwa\_gatunku.
6. **karta\_szczepienia:** Informacje na temat szczepień dla zwierząt, z id\_karta\_szczepienia, nazwa\_szczepienia, data\_ostatniego\_szczepienia, i zwierze\_id\_zwierze.
7. **lek:** Informacje o lekach, z id\_lek, nazwa\_leku, i choroba\_id\_choroba.
8. **lekarz:** Dane na temat lekarzy weterynarii, z id\_lekarz, imię, nazwisko, data\_zatrudnienia, godzina\_rozporozczenia\_pracy, godzina\_zakończenia\_pracy, i specjalizacja\_id\_specjalizacja.
9. **rodzaj:** Informacje o rodzajach zwierząt, z id\_rodzaj, nazwa\_rodzaju, i gatunek\_id\_gatunek.
10. **specjalizacja:** Dane na temat specjalizacji lekarzy, z id\_specjalizacja i nazwa\_specjalizacji.
11. **wizyta:** Informacje o wizytach lekarskich dla zwierząt, z id\_wizyta, data\_, godzina, zwierze\_id\_zwierze, lekarz\_id\_lekarz, i gabinet\_id\_gabinet.
12. **właściciel:** Dane na temat właścicieli zwierząt, z id\_właściciel, imię, nazwisko, ulica, numer\_domu, numer\_mieszkania, i adres\_id\_adres.
13. **zabieg:** Informacje o zabiegach wykonywanych podczas wizyt, z id\_zabieg, nazwa\_zabiegu, i wizyta\_id\_wizyta.
14. **zwierze:** Dane na temat zwierząt, z id\_zwierze, imię, data\_urodzenia, płeć, umaszczenie, gatunek\_id\_gatunek, i właściciel\_id\_właściciel.

### 3.1. Kod SQL

```
CREATE TABLE adres (  
    id_adres    NUMBER(10) NOT NULL,  
    kod_pocztowy VARCHAR2(11),  
    miasto      VARCHAR2(20)  
);
```

```
ALTER TABLE adres ADD CONSTRAINT adres_pk PRIMARY KEY (id_adres);
```

```
CREATE TABLE choroba (  
    id_choroba  NUMBER(10) NOT NULL,  
    nazwa_choroby VARCHAR2(30)  
);
```

```
ALTER TABLE choroba ADD CONSTRAINT choroba_pk PRIMARY KEY (id_choroba);
```

```
CREATE TABLE diagnoza (  
    wizyta_id_wizyta  NUMBER(10) NOT NULL,  
    choroba_id_choroba NUMBER(10) NOT NULL  
);
```

```
ALTER TABLE diagnoza ADD CONSTRAINT diagnoza_pk PRIMARY KEY (wizyta_id_wizyta,  
    choroba_id_choroba);
```

```
CREATE TABLE gabinet (  
    id_gabinet  NUMBER(10) NOT NULL,  
    numer_gabinetu NUMBER  
);
```

```
ALTER TABLE gabinet ADD CONSTRAINT gabinet_pk PRIMARY KEY (id_gabinet);
```



```
CREATE TABLE gatunek (
```

```
    id_gatunek    NUMBER(10) NOT NULL,
```

```
    nazwa_gatunku VARCHAR2(20)
```

```
);
```

```
ALTER TABLE gatunek ADD CONSTRAINT gatunek_pk PRIMARY KEY (id_gatunek);
```

```
CREATE TABLE karta_szczepienia (
```

```
    id_karta_szczepienia    NUMBER(10) NOT NULL,
```

```
    nazwa_szczepienia       VARCHAR2(35),
```

```
    data_ostatniego_szczepienia DATE,
```

```
    zwierzę_id_zwierzę      NUMBER(10)
```

```
);
```

```
ALTER TABLE karta_szczepienia ADD CONSTRAINT karta_szczepienia_pk PRIMARY KEY  
(id_karta_szczepienia);
```

```
CREATE TABLE lek (
```

```
    id_lek        NUMBER(10) NOT NULL,
```

```
    nazwa_leku     VARCHAR2(20),
```

```
    choroba_id_choroba NUMBER(10)
```

```
);
```

```
ALTER TABLE lek ADD CONSTRAINT lek_pk PRIMARY KEY (id_lek);
```

```
CREATE TABLE lekarz (  
    id_lekarz          NUMBER(10) NOT NULL,  
    imię              VARCHAR2(20),  
    nazwisko           VARCHAR2(20),  
    data_zatrudnienia  DATE,  
    godzina_rozpoczęcia_pracy  DATE,  
    godzina_zakończenia_pracy  DATE,  
    specjalizacja_id_specjalizacja NUMBER(10) NOT NULL  
);
```

```
ALTER TABLE lekarz ADD CONSTRAINT lekarz_pk PRIMARY KEY (id_lekarz);
```

```
CREATE TABLE rodzaj (  
    id_rodzaj          NUMBER(10) NOT NULL,  
    nazwa_rodzaju      VARCHAR2(20),  
    gatunek_id_gatunek NUMBER(10) NOT NULL  
);
```

```
ALTER TABLE rodzaj ADD CONSTRAINT rodzaj_pk PRIMARY KEY (id_rodzaj);
```

```
CREATE TABLE specjalizacja (  
    id_specjalizacja  NUMBER(10) NOT NULL,  
    nazwa_specjalizacji VARCHAR2(40)  
);
```

```
ALTER TABLE specjalizacja ADD CONSTRAINT specjalizacja_pk PRIMARY KEY (id_specjalizacja);
```

```
CREATE TABLE wizyta (
```

```
    id_wizyta      NUMBER(10) NOT NULL,
```

```
    data_          DATE,
```

```
    godzina        DATE,
```

```
    zwierze_id_zwierze NUMBER(10),
```

```
    lekarz_id_lekarz  NUMBER(10) NOT NULL,
```

```
    gabinet_id_gabinet NUMBER(10)
```

```
);
```

```
ALTER TABLE wizyta ADD CONSTRAINT wizyta_pk PRIMARY KEY (id_wizyta);
```

```
CREATE TABLE właściciel (
```

```
    id_właściciel  NUMBER(10) NOT NULL,
```

```
    imię           VARCHAR2(20),
```

```
    nazwisko       VARCHAR2(20),
```

```
    ulica          VARCHAR2(20),
```

```
    numer_domu     NUMBER,
```

```
    numer_mieszkania NUMBER,
```

```
    adres_id_adres  NUMBER(10) NOT NULL
```

```
);
```

```
ALTER TABLE właściciel ADD CONSTRAINT właściciel_pk PRIMARY KEY (id_właściciel);
```

```
CREATE TABLE zabieg (
```

```
    id_zabieg      NUMBER(10) NOT NULL,
```

```
    nazwa_zabiegu  VARCHAR2(40),
```

```
    wizyta_id_wizyta NUMBER(10)
```

```
);
```

```
ALTER TABLE zabieg ADD CONSTRAINT zabieg_pk PRIMARY KEY (id_zabieg);
```

```
CREATE TABLE zwierzę (  
    id_zwierzę          NUMBER(10) NOT NULL,  
    imię                VARCHAR2(20),  
    data_urodzenia      DATE,  
    płeć                VARCHAR2(6),  
    umaszczenie         VARCHAR2(20),  
    gatunek_id_gatunek  NUMBER(10) NOT NULL,  
    właściciel_id_właściciel NUMBER(10) NOT NULL  
);
```

```
ALTER TABLE zwierzę ADD CONSTRAINT zwierzę_pk PRIMARY KEY (id_zwierzę);
```

```
ALTER TABLE diagnoza  
    ADD CONSTRAINT diagnoza_choroba_fk FOREIGN KEY (choroba_id_choroba)  
    REFERENCES choroba (id_choroba);
```

```
ALTER TABLE diagnoza  
    ADD CONSTRAINT diagnoza_wizyta_fk FOREIGN KEY (wizyta_id_wizyta)  
    REFERENCES wizyta (id_wizyta);
```

```
ALTER TABLE karta_szczepienia  
    ADD CONSTRAINT karta_szczepienia_zwierzę_fk FOREIGN KEY (zwierzę_id_zwierzę)  
    REFERENCES zwierzę (id_zwierzę);
```

```
ALTER TABLE lek  
    ADD CONSTRAINT lek_choroba_fk FOREIGN KEY (choroba_id_choroba)  
    REFERENCES choroba (id_choroba);
```

```
ALTER TABLE lekarz
```

```
ADD CONSTRAINT lekarz_specjalizacja_fk FOREIGN KEY (specjalizacja_id_specjalizacja)
REFERENCES specjalizacja (id_specjalizacja);
```

```
ALTER TABLE rodzaj
```

```
ADD CONSTRAINT rodzaj_gatunek_fk FOREIGN KEY (gatunek_id_gatunek)
REFERENCES gatunek (id_gatunek);
```

```
ALTER TABLE wizyta
```

```
ADD CONSTRAINT wizyta_gabinet_fk FOREIGN KEY (gabinet_id_gabinet)
REFERENCES gabinet (id_gabinet);
```

```
ALTER TABLE wizyta
```

```
ADD CONSTRAINT wizyta_lekarz_fk FOREIGN KEY (lekarz_id_lekarz)
REFERENCES lekarz (id_lekarz);
```

```
ALTER TABLE wizyta
```

```
ADD CONSTRAINT wizyta_zwierze_fk FOREIGN KEY (zwierze_id_zwierze)
REFERENCES zwierze (id_zwierze);
```

```
ALTER TABLE wlasiciel
```

```
ADD CONSTRAINT wlasiciel_adres_fk FOREIGN KEY (adres_id_adres)
REFERENCES adres (id_adres);
```

```
ALTER TABLE zabieg
```

```
ADD CONSTRAINT zabieg_wizyta_fk FOREIGN KEY (wizyta_id_wizyta)
REFERENCES wizyta (id_wizyta);
```

ALTER TABLE zwierzę

ADD CONSTRAINT zwierzę\_gatunek\_fk FOREIGN KEY (gatunek\_id\_gatunek)  
REFERENCES gatunek (id\_gatunek);

ALTER TABLE zwierzę

ADD CONSTRAINT zwierzę\_właściciel\_fk FOREIGN KEY (właściciel\_id\_właściciel)  
REFERENCES właściciel (id\_właściciel);

INSERT INTO adres (id\_adres, kod\_pocztowy, miasto)

VALUES

(1, '00-001', 'Warszawa'),  
(2, '50-123', 'Wrocław'),  
(3, '80-456', 'Gdańsk'),  
(4, '02-789', 'Kraków'),  
(5, '90-234', 'Poznań'),  
(6, '70-567', 'Szczecin'),  
(7, '20-890', 'Łódź'),  
(8, '30-345', 'Katowice'),  
(9, '40-678', 'Gdynia'),  
(10, '60-901', NULL);

INSERT INTO gatunek (id\_gatunek, nazwa\_gatunku)

VALUES

(1, 'Kot domowy'),  
(2, 'Pies buldog'),  
(3, 'Chomik syryjski'),  
(4, 'Ptak egzotyczny'),  
(5, 'Królik hodowlany'),  
(6, 'Żółw lądowy'),  
(7, 'Ryba akwariowa'),  
(8, 'Jeż europejski'),

(9, 'Wiewiórka szara'),

(10, 'Legwan zielony');

INSERT INTO specjalizacja (id\_specjalizacja, nazwa\_specjalizacji)

VALUES

(1, 'Choroby psów i kotów'),

(2, 'Chirurgia weterynaryjna'),

(3, 'Choroby zwierząt futerkowych'),

(4, 'Choroby drobiu oraz ptaków ozdobnych'),

(5, 'Rozród zwierząt'),

(6, 'Choroby przeżuwaczy'),

(7, 'Choroby ryb'),

(8, 'Chirurgia weterynaryjna'),

(9, 'Radiologia weterynaryjna'),

(10, 'Choroby zwierząt nieudomowionych');

INSERT INTO choroba (id\_choroba, nazwa\_choroby)

VALUES

(1, 'Zespół FLUTD'),

(2, 'Zapalenie gardła'),

(3, 'Choroba mokrego ogona'),

(4, 'Ornitoza'),

(5, 'Anaplazmoza'),

(6, 'Reumatyzm'),

(7, 'Choroba serca'),

(8, 'Choroba tarczycy'),

(9, 'Nowotwór'),

(10, 'Grzybica');

INSERT INTO lek (id\_lek, nazwa\_leku, choroba\_id\_choroba)

VALUES

(1, 'Paracetamol', 1),  
(2, 'Ibuprofen', 2),  
(3, 'Antyhistaminik', 3),  
(4, 'Diseptol', 4),  
(5, 'Ventolin', 5),  
(6, 'Metotreksat', 6),  
(7, 'Bentoks', 7),  
(8, 'Eutiroks', 8),  
(9, 'Chemioterapia', 9),  
(10, 'Lentyrox', 10);

INSERT INTO rodzaj (id\_rodzaj, nazwa\_rodzaju, gatunek\_id\_gatunek)

VALUES

(1, 'Kot', 1),  
(2, 'Pies', 2),  
(3, 'Chomik', 3),  
(4, 'Ptak', 4),  
(5, 'Królik', 5),  
(6, 'Żółw', 6),  
(7, 'Ryba', 7),  
(8, 'Jeź', 8),  
(9, 'Wiewiórka', 9),  
(10, 'Legwan', 10);



```
INSERT INTO gabinet (id_gabinet, numer_gabinetu)
```

```
VALUES
```

```
(1, 101),  
(2, 202),  
(3, 303),  
(4, 404),  
(5, 505),  
(6, 606),  
(7, 707),  
(8, 808),  
(9, 909),  
(10, 1010);
```

```
INSERT INTO właściciel (id_właściciel, imię, nazwisko, ulica, numer_domu, numer_mieszkania,  
adres_id_adres)
```

```
VALUES
```

```
(1, 'Adam', 'Nowak', 'Kwiatowa', 5, 3, 1),  
(2, 'Ewa', 'Kowalska', 'Słoneczna', 12, 8, 2),  
(3, 'Michał', 'Wiśniewski', NULL, 7, 2, 3),  
(4, 'Anna', 'Dąbrowska', 'Polna', 20, 15, 4),  
(5, 'Piotr', 'Lis', 'Klonowa', 15, 7, 5),  
(6, 'Karolina', 'Jankowska', 'Różana', 10, 5, 6),  
(7, 'Mateusz', 'Wójcik', 'Żeromskiego', 3, 1, 7),  
(8, 'Aleksandra', 'Pawlak', 'Piękna', 18, 10, 8),  
(9, 'Grzegorz', 'Flis', 'Kwiatowa', 25, 12, 9),  
(10, 'Monika', 'Kaczmarek', 'Słowackiego', 8, 4, 10);
```

```
INSERT INTO zwierzę (id_zwierzę, imię, data_urodzenia, płeć, umaszczenie, gatunek_id_gatunek,
właściciel_id_właściciel)
```

```
VALUES
```

```
(1, 'Burek', TO_DATE('2018-05-15', 'YYYY-MM-DD'), 'Samiec', 'Brązowe', 1, 1),
(2, 'Luna', TO_DATE('2019-02-28', 'YYYY-MM-DD'), 'Samica', 'Czarne', 2, 2),
(3, 'Max', TO_DATE('2017-09-10', 'YYYY-MM-DD'), 'Samiec', 'Białe', 3, 3),
(4, 'Mia', TO_DATE('2020-07-03', 'YYYY-MM-DD'), 'Samica', NULL, 1, 4),
(5, 'Rocky', TO_DATE('2016-12-20', 'YYYY-MM-DD'), 'Samiec', 'Czekoladowe', 2, 5),
(6, 'Daisy', TO_DATE('2019-11-05', 'YYYY-MM-DD'), 'Samica', 'Białe z cętkami', 3, 6),
(7, 'Charlie', TO_DATE('2018-04-12', 'YYYY-MM-DD'), 'Samiec', 'Czarno-Białe', 1, 7),
(8, 'Zoe', TO_DATE('2017-08-08', 'YYYY-MM-DD'), 'Samica', 'Szare', 2, 8),
(9, 'Oscar', TO_DATE('2020-01-25', 'YYYY-MM-DD'), 'Samiec', 'Czarno-Bure', 3, 9),
(10, 'Molly', TO_DATE('2016-06-30', 'YYYY-MM-DD'), 'Samica', 'Brązowe', 1, 10);
```

```
INSERT INTO lekarz (id_lekarz, imię, nazwisko, data_zatrudnienia, godzina_rozpoczęcia_pracy,
godzina_zakończenia_pracy, specjalizacja_id_specjalizacja)
```

```
VALUES
```

```
(1, 'Jan', 'Kowalski', NULL, TO_DATE('08:00:00', 'HH24:MI:SS'), TO_DATE('16:00:00', 'HH24:MI:SS'),
1),
(2, 'Anna', 'Nowak', TO_DATE('2019-06-15', 'YYYY-MM-DD'), TO_DATE('09:30:00', 'HH24:MI:SS'),
TO_DATE('17:30:00', 'HH24:MI:SS'), 2),
(3, 'Piotr', 'Wiśniewski', TO_DATE('2021-01-10', 'YYYY-MM-DD'), TO_DATE('08:30:00',
'HH24:MI:SS'), TO_DATE('16:30:00', 'HH24:MI:SS'), 3),
(4, 'Magdalena', 'Dąbrowska', TO_DATE('2022-02-20', 'YYYY-MM-DD'), TO_DATE('10:00:00',
'HH24:MI:SS'), TO_DATE('18:00:00', 'HH24:MI:SS'), 4),
(5, 'Marcin', 'Lis', TO_DATE('2018-08-05', 'YYYY-MM-DD'), TO_DATE('07:30:00', 'HH24:MI:SS'),
TO_DATE('15:30:00', 'HH24:MI:SS'), 5),
(6, 'Karolina', 'Jankowska', TO_DATE('2020-11-12', 'YYYY-MM-DD'), TO_DATE('11:00:00',
'HH24:MI:SS'), TO_DATE('19:00:00', 'HH24:MI:SS'), 6),
(7, 'Mateusz', 'Wójcik', TO_DATE('2019-04-02', 'YYYY-MM-DD'), TO_DATE('08:00:00',
'HH24:MI:SS'), TO_DATE('16:00:00', 'HH24:MI:SS'), 7),
(8, 'Aleksandra', 'Pawlak', TO_DATE('2021-07-25', 'YYYY-MM-DD'), TO_DATE('09:30:00',
'HH24:MI:SS'), TO_DATE('17:30:00', 'HH24:MI:SS'), 8),
(9, 'Grzegorz', 'Nowicki', TO_DATE('2017-12-15', 'YYYY-MM-DD'), TO_DATE('08:30:00',
'HH24:MI:SS'), TO_DATE('16:30:00', 'HH24:MI:SS'), 9),
```

```
(10, 'Monika', 'Kaczmarek', TO_DATE('2022-04-18', 'YYYY-MM-DD'), TO_DATE('10:00:00', 'HH24:MI:SS'), TO_DATE('18:00:00', 'HH24:MI:SS'), 10);
```

```
INSERT INTO wizyta (id_wizyta, data_, godzina, zwierze_id_zwierze, lekarz_id_lekarz, gabinet_id_gabinet)  
VALUES
```

```
(1, TO_DATE('2023-01-15', 'YYYY-MM-DD'), TO_DATE('10:00:00', 'HH24:MI:SS'), 1, 1, 1),  
(2, TO_DATE('2022-11-10', 'YYYY-MM-DD'), TO_DATE('11:30:00', 'HH24:MI:SS'), 2, 2, 2),  
(3, TO_DATE('2023-03-20', 'YYYY-MM-DD'), TO_DATE('13:15:00', 'HH24:MI:SS'), 3, 3, 3),  
(4, TO_DATE('2022-09-25', 'YYYY-MM-DD'), TO_DATE('14:45:00', 'HH24:MI:SS'), 4, 4, 4),  
(5, TO_DATE('2023-05-15', 'YYYY-MM-DD'), TO_DATE('16:30:00', 'HH24:MI:SS'), 5, 5, 5),  
(6, TO_DATE('2022-08-12', 'YYYY-MM-DD'), TO_DATE('09:00:00', 'HH24:MI:SS'), 6, 6, 6),  
(7, TO_DATE('2023-02-28', 'YYYY-MM-DD'), TO_DATE('10:45:00', 'HH24:MI:SS'), 7, 7, 7),  
(8, TO_DATE('2022-12-15', 'YYYY-MM-DD'), TO_DATE('12:30:00', 'HH24:MI:SS'), 8, 8, 8),  
(9, TO_DATE('2023-04-25', 'YYYY-MM-DD'), TO_DATE('14:15:00', 'HH24:MI:SS'), 9, 9, 9),  
(10, TO_DATE('2022-10-30', 'YYYY-MM-DD'), TO_DATE('15:45:00', 'HH24:MI:SS'), 10, 10, 10);
```

```
INSERT INTO diagnoza (wizyta_id_wizyta, choroba_id_choroba)
```

```
VALUES
```

```
(1, 1),  
(2, 2),  
(3, 3),  
(4, 4),  
(5, 5),  
(6, 6),  
(7, 7),  
(8, 8),  
(9, 9),  
(10, 10);
```

```
INSERT INTO karta_szczepienia (id_karta_szczepienia, nazwa_szczepienia, data_ostatniego_szczepienia,  
zwierze_id_zwierze)
```

```
VALUES
```

(1, 'Szczepienie przeciwko wściekliźnie', TO\_DATE('2023-01-10', 'YYYY-MM-DD'), 1),  
 (2, 'Szczepienie przeciwko chorobie zakaźnej', TO\_DATE('2022-11-05', 'YYYY-MM-DD'), 2),  
 (3, 'Szczepienie przeciwko parwowirozie', TO\_DATE('2023-03-15', 'YYYY-MM-DD'), 3),  
 (4, 'Szczepienie przeciwko ptasiej grypie', TO\_DATE('2022-09-20', 'YYYY-MM-DD'), 4),  
 (5, 'Szczepienie przeciwko królikołąpie', TO\_DATE('2023-05-12', 'YYYY-MM-DD'), 5),  
 (6, 'Szczepienie przeciwko grzybicy skóry', TO\_DATE('2022-08-08', 'YYYY-MM-DD'), 6),  
 (7, 'Szczepienie przeciwko wirusowi rybniej ospie', TO\_DATE('2023-02-28', 'YYYY-MM-DD'), 7),  
 (8, 'Szczepienie przeciwko jeżowym kleszczom', TO\_DATE('2022-12-10', 'YYYY-MM-DD'), 8),  
 (9, 'Szczepienie przeciwko wiewiórczej ospie', TO\_DATE('2023-04-18', 'YYYY-MM-DD'), 9),  
 (10, 'Szczepienie przeciwko bakterii legwany', TO\_DATE('2022-10-25', 'YYYY-MM-DD'), 10);

INSERT INTO zabieg (id\_zabieg, nazwa\_zabiegu, wizyta\_id\_wizyta)

VALUES

(1, 'Badanie krwi', 1),  
 (2, 'Założenie gipsu', 2),  
 (3, 'Zabieg chirurgiczny', 3),  
 (4, 'Nakładanie szyny', 4),  
 (5, 'Poród', 5),  
 (6, 'Rehabilitacja po urazie', 6),  
 (7, 'Badanie skrzel', 7),  
 (8, 'Zabieg onkologiczny', 8),  
 (9, 'Badanie radiologiczne', 9),  
 (10, 'Szczepienie przeciwko chorobie zakaźnej', 10);

### 3.2. Kwerendy

#### 1. Wybierz wszystkie zwierzęta i ich właścicieli:

```
SELECT z.imię AS "Imię Zwierzęcia", z.data_urodzenia, z.płeć, z.umaszczenie,
       g.nazwa_gatunku AS "Gatunek", w.imię || ' ' || w.nazwisko AS "Właściciel"
FROM zwierzę z
JOIN gatunek g ON z.gatunek_id_gatunek = g.id_gatunek
```

JOIN właściciel w ON z.właściciel\_id\_właściciel = w.id\_właściciel;

```
361 SELECT z.imię AS "Imię Zwierzęcia", z.data_urodzenia, z.płeć, z.umaszczenie,  
362       g.nazwa_gatunku AS "Gatunek", w.imię || ' ' || w.nazwisko AS "Właściciel"  
363 FROM zwierzę z  
364 JOIN gatunek g ON z.gatunek_id_gatunek = g.id_gatunek  
365 JOIN właściciel w ON z.właściciel_id_właściciel = w.id_właściciel;  
367
```

Script Output x Query Result x						
SQL   All Rows Fetched: 10 in 0.163 seconds						
	Imię Zwierzęcia	DATA_URODZENIA	PŁEĆ	UMASZCZENIE	Gatunek	Właściciel
1	Burek	15-MAY-18	Samiec Brązowe		Kot domowy	Adam Nowak
2	Luna	28-FEB-19	Samica Czarne		Pies buldog	Ewa Kowalska
3	Max	10-SEP-17	Samiec Białe		Chomik syryjski	Michał Wiśniewski
4	Mia	03-JUL-20	Samica (null)		Kot domowy	Anna Dąbrowska
5	Rocky	20-DEC-16	Samiec Czekoladowe		Pies buldog	Piotr Lis
6	Daisy	05-NOV-19	Samica Białe z cętkami		Chomik syryjski	Karolina Jankowska
7	Charlie	12-APR-18	Samiec Czarno-Białe		Kot domowy	Mateusz Wójcik
8	Zoe	08-AUG-17	Samica Szare		Pies buldog	Aleksandra Pawlak
9	Oscar	25-JAN-20	Samiec Czarno-Bure		Chomik syryjski	Grzegorz Flis
10	Molly	30-JUN-16	Samica Brązowe		Kot domowy	Monika Kaczmarek

## 2. Wybierz zwierzęta, które są młodsze niż 5 lat:

SELECT \* FROM zwierzę

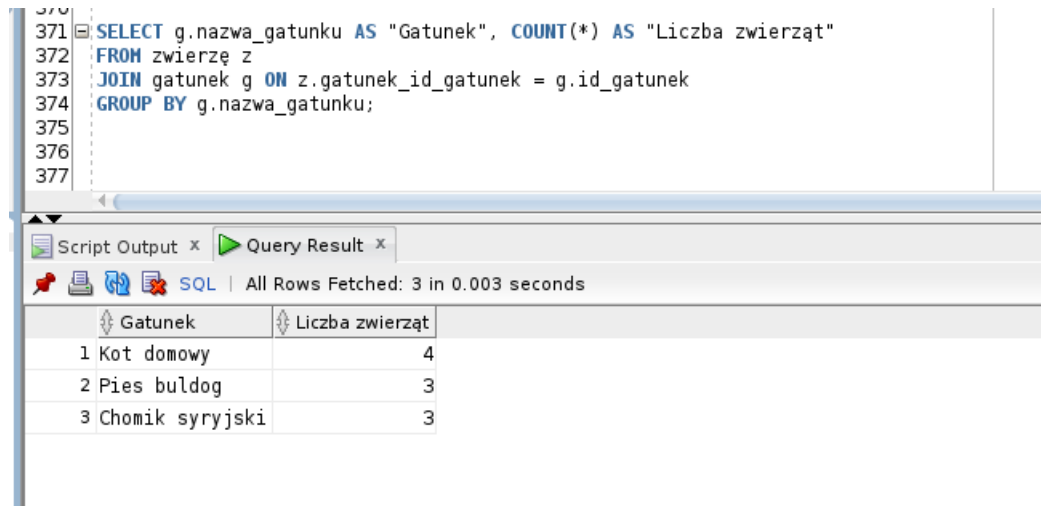
WHERE EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM data\_urodzenia) < 5;

```
368 SELECT * FROM zwierzę  
369 WHERE EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM data_urodzenia) < 5;  
370
```

Script Output x Query Result x						
SQL   All Rows Fetched: 2 in 0.024 seconds						
	ID_ZWIERZĘ	IMIĘ	DATA_URODZENIA	PŁEĆ	UMASZCZENIE	GATUNEK_ID_GATUNEK
1	4	Mia	03-JUL-20	Samica (null)		1
2	9	Oscar	25-JAN-20	Samiec Czarno-Bure		3

### 3. Znajdź liczbę zwierząt dla każdego gatunku:

```
SELECT g.nazwa_gatunku AS "Gatunek", COUNT(*) AS "Liczba zwierząt"
FROM zwierzę z
JOIN gatunek g ON z.gatunek_id_gatunek = g.id_gatunek
GROUP BY g.nazwa_gatunku;
```



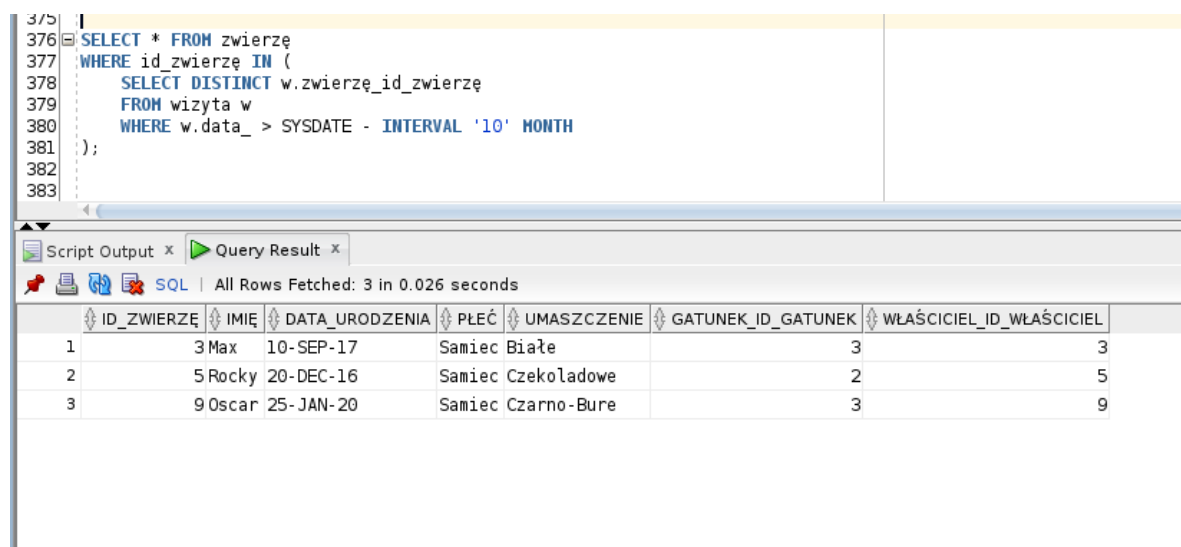
Script Output x Query Result x

SQL | All Rows Fetched: 3 in 0.003 seconds

Gatunek	Liczba zwierząt
1 Kot domowy	4
2 Pies buldog	3
3 Chomik syryjski	3

### 4. Znajdź zwierzęta, które miały wizytę w ostatnich 10 miesiącach:

```
SELECT * FROM zwierzę
WHERE id_zwierzę IN (
    SELECT DISTINCT w.zwierzę_id_zwierzę
    FROM wizyta w
    WHERE w.data_ > SYSDATE - INTERVAL '10' MONTH
);
```



Script Output x Query Result x

SQL | All Rows Fetched: 3 in 0.026 seconds

ID_ZWIERZĘ	IMIĘ	DATA_URODZENIA	PLEĆ	UMASZCZENIE	GATUNEK_ID_GATUNEK	WŁAŚCICIEL_ID_WŁAŚCICIEL
1	3 Max	10-SEP-17	Samiec	Białe	3	3
2	5 Rocky	20-DEC-16	Samiec	Czekoladowe	2	5
3	9 Oscar	25-JAN-20	Samiec	Czarno-Bure	3	9

### 5. Znajdź liczbę wizyt dla każdego lekarza w danym roku:

```
SELECT EXTRACT(YEAR FROM w.data_) AS "Rok", l.imię || ' ' || l.nazwisko AS "Lekarz",
```

```

COUNT(*) AS "Liczba wizyt"

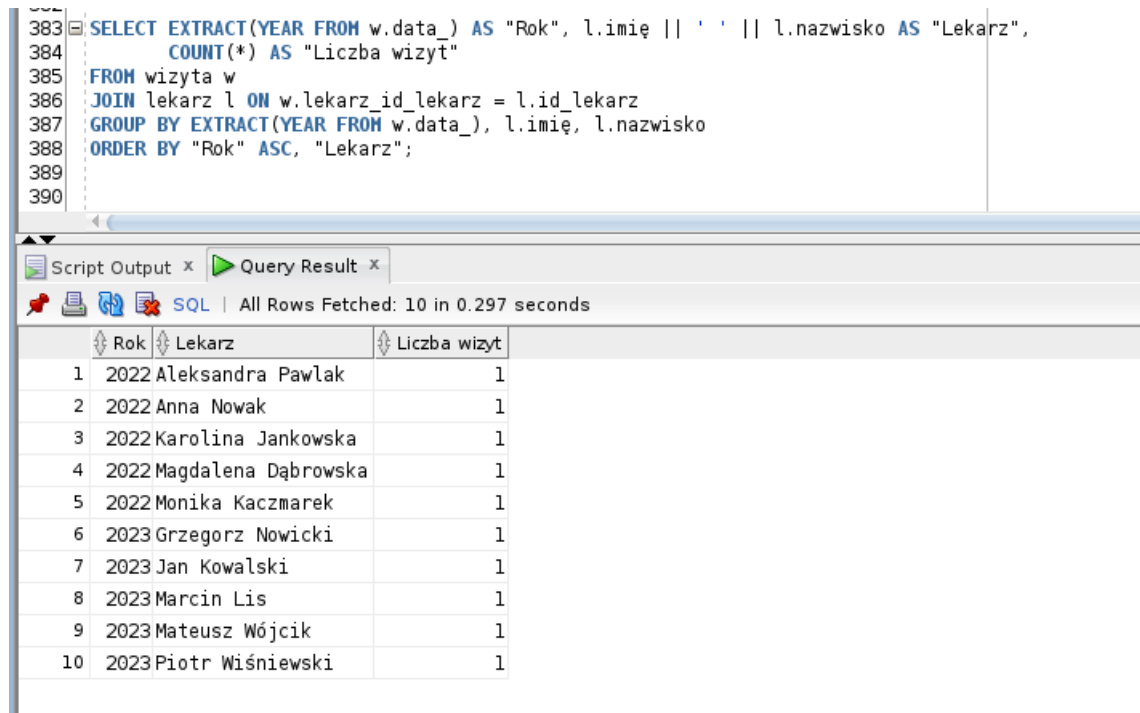
FROM wizyta w

JOIN lekarz l ON w.lekarz_id_lekarz = l.id_lekarz

GROUP BY EXTRACT(YEAR FROM w.data_), l.imię, l.nazwisko

ORDER BY "Rok" ASC, "Lekarz";

```



```

383 SELECT EXTRACT(YEAR FROM w.data_) AS "Rok", l.imię || ' ' || l.nazwisko AS "Lekarz",
384         COUNT(*) AS "Liczba wizyt"
385 FROM wizyta w
386 JOIN lekarz l ON w.lekarz_id_lekarz = l.id_lekarz
387 GROUP BY EXTRACT(YEAR FROM w.data_), l.imię, l.nazwisko
388 ORDER BY "Rok" ASC, "Lekarz";
389
390

```

	Rok	Lekarz	Liczba wizyt
1	2022	Aleksandra Pawlak	1
2	2022	Anna Nowak	1
3	2022	Karolina Jankowska	1
4	2022	Magdalena Dąbrowska	1
5	2022	Monika Kaczmarek	1
6	2023	Grzegorz Nowicki	1
7	2023	Jan Kowalski	1
8	2023	Marcin Lis	1
9	2023	Mateusz Wójcik	1
10	2023	Piotr Wiśniewski	1

**6. Identyfikacja najnowszej wizyty dla każdej samicy zwierzęcia na podstawie daty wizyty, przypisując numer wizyty w kolejności malejącej daty:**

```

SELECT * FROM (

    SELECT zwierzę.id_zwierzę, zwierzę.imię, wizyta.data_,

        ROW_NUMBER() OVER (PARTITION BY zwierzę.id_zwierzę ORDER BY wizyta.data_
DESC) AS "Numer Wizyty"

    FROM zwierzę

    JOIN wizyta ON zwierzę.id_zwierzę = wizyta.zwierzę_id_zwierzę

    WHERE zwierzę.płeć = 'Samica'

) T

WHERE "Numer Wizyty" = 1;

```

```

390 SELECT * FROM (
391     SELECT zwierzę.id_zwierzę, zwierzę.imię, wizyta.data_,
392           ROW_NUMBER() OVER (PARTITION BY zwierzę.id_zwierzę ORDER BY wizyta.data_ DESC) AS "Numer Wizyty"
393     FROM zwierzę
394     JOIN wizyta ON zwierzę.id_zwierzę = wizyta.zwierzę_id_zwierzę
395     WHERE zwierzę.płeć = 'Samica'
396 ) T
397 WHERE "Numer Wizyty" = 1;
398
399

```

ID_ZWIERZĘ	IMIĘ	DATA_	Numer Wizyty
1	2 Luna	10-NOV-22	1
2	4 Mia	25-SEP-22	1
3	6 Daisy	12-AUG-22	1
4	8 Zoe	15-DEC-22	1
5	10 Molly	30-OCT-22	1

## 7. Wizyty w weekendy w miesiącach od kwietnia do czerwca, obsługiwane przez jednego lekarza:

```
SELECT id_wizyta, data_, godzina,
```

```
CASE
```

```
    WHEN TO_CHAR(data_, 'DY') = 'SAT' THEN 'Weekend'
```

```
    WHEN TO_CHAR(data_, 'DY') = 'SUN' THEN 'Weekend'
```

```
    ELSE 'Dzień roboczy'
```

```
END AS "Typ Dnia"
```

```
FROM wizyta
```

```
WHERE TO_CHAR(data_, 'YYYY-MM') IN (
```

```
    SELECT TO_CHAR(data_, 'YYYY-MM')
```

```
    FROM wizyta
```

```
    GROUP BY TO_CHAR(data_, 'YYYY-MM')
```

```
    HAVING COUNT(DISTINCT lekarz_id_lekarz) = 1
```

```
) AND EXTRACT(MONTH FROM data_) BETWEEN 4 AND 6;
```



```

399 SELECT id_wizyta, data_, godzina,
400 CASE
401     WHEN TO_CHAR(data_, 'DY') = 'SAT' THEN 'Weekend'
402     WHEN TO_CHAR(data_, 'DY') = 'SUN' THEN 'Weekend'
403     ELSE 'Dzień roboczy'
404 END AS "Typ Dnia"
405 FROM wizyta
406 WHERE TO_CHAR(data_, 'YYYY-MM') IN (
407     SELECT TO_CHAR(data_, 'YYYY-MM')
408     FROM wizyta
409     GROUP BY TO_CHAR(data_, 'YYYY-MM')
410     HAVING COUNT(DISTINCT lekarz_id_lekarz) = 1
411 ) AND EXTRACT(MONTH FROM data_) BETWEEN 4 AND 6;
412
413

```

Script Output x Query Result x

SQL All Rows Fetched: 2 in 0.025 seconds

ID_WIZYTA	DATA	GODZINA	Typ Dnia
1	5 15-MAY-23	01-JAN-24	Dzień roboczy
2	9 25-APR-23	01-JAN-24	Dzień roboczy

## 8. Kategorie wiekowe dla zwierząt:

SELECT id\_zwierzę, imię, data\_urodzenia,

CASE

WHEN MONTHS\_BETWEEN(SYSDATE, data\_urodzenia) < 12 THEN 'Młode'

WHEN MONTHS\_BETWEEN(SYSDATE, data\_urodzenia) >= 12 AND MONTHS\_BETWEEN(SYSDATE, data\_urodzenia) < 60 THEN 'Dorosłe'

ELSE 'Starsze'

END AS "Kategoria Wiekowa"

FROM zwierzę

WHERE data\_urodzenia IS NOT NULL

ORDER BY "Kategoria Wiekowa" DESC, data\_urodzenia ASC;

```

413 SELECT id_zwierzę, imię, data_urodzenia,
414 CASE
415     WHEN MONTHS_BETWEEN(SYSDATE, data_urodzenia) < 12 THEN 'Młode'
416     WHEN MONTHS_BETWEEN(SYSDATE, data_urodzenia) >= 12 AND MONTHS_BETWEEN(SYSDATE, data_urodzenia) < 60 THEN 'Dorosłe'
417     ELSE 'Starsze'
418 END AS "Kategoria Wiekowa"
419 FROM zwierzę
420 WHERE data_urodzenia IS NOT NULL
421 ORDER BY "Kategoria Wiekowa" DESC, data_urodzenia ASC;
422
423

```

Script Output x Query Result x

SQL All Rows Fetched: 10 in 0.035 seconds

ID_ZWIERZE	IMIĘ	DATA_URODZENIA	Kategoria Wiekowa
1	10 Molly	30-JUN-16	Starsze
2	5 Rocky	20-DEC-16	Starsze
3	8 Zoe	08-AUG-17	Starsze
4	3 Max	10-SEP-17	Starsze
5	7 Charlie	12-APR-18	Starsze
6	1 Burek	15-MAY-18	Starsze
7	2 Luna	28-FEB-19	Dorosłe
8	6 Daisy	05-NOV-19	Dorosłe
9	9 Oscar	25-JAN-20	Dorosłe
10	4 Mia	03-JUL-20	Dorosłe

## 9. Wyświetlanie szczegółów wizyt, wliczając liczbę diagnoz, średni wiek zwierzęcia i datę ostatniego szczepienia:

SELECT

id\_wizyta,

data\_,

godzina,

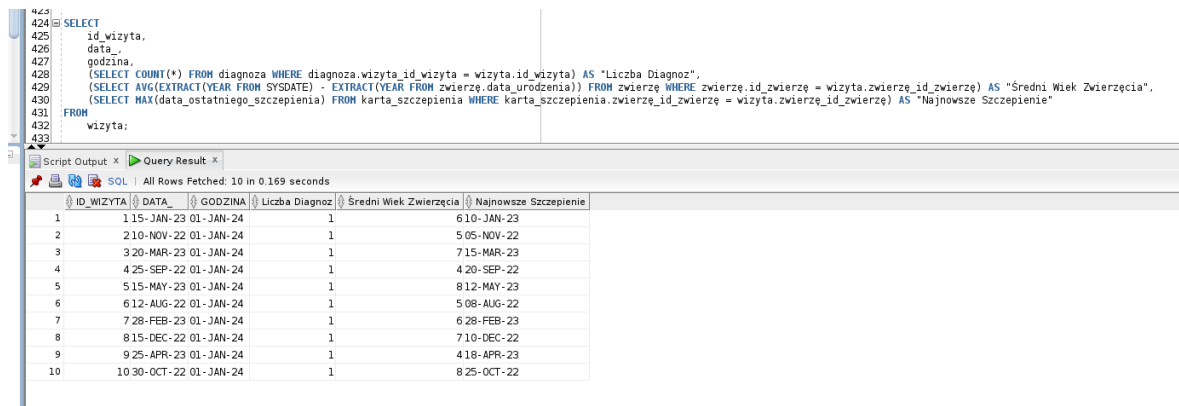
(SELECT COUNT(\*) FROM diagnoza WHERE diagnoza.wizyta\_id\_wizyta = wizyta.id\_wizyta)  
AS "Liczba Diagnoz",

(SELECT AVG(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM  
zwierzę.data\_urodzenia)) FROM zwierzę WHERE zwierzę.id\_zwierzę =  
wizyta.zwierzę\_id\_zwierzę) AS "Średni Wiek Zwierzęcia",

(SELECT MAX(data\_ostatniego\_szczepienia) FROM karta\_szczepienia WHERE  
karta\_szczepienia.zwierzę\_id\_zwierzę = wizyta.zwierzę\_id\_zwierzę) AS "Najnowsze Szczepienie"

FROM

wizyta;



The screenshot shows a SQL query in a text editor and its results in a table. The query is as follows:

```
SELECT
    id_wizyta,
    data_,
    godzina,
    (SELECT COUNT(*) FROM diagnoza WHERE diagnoza.wizyta_id_wizyta = wizyta.id_wizyta) AS "Liczba Diagnoz",
    (SELECT AVG(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM zwierzę.data_urodzenia)) FROM zwierzę WHERE zwierzę.id_zwierzę = wizyta.zwierzę_id_zwierzę) AS "Średni Wiek Zwierzęcia",
    (SELECT MAX(data_ostatniego_szczepienia) FROM karta_szczepienia WHERE karta_szczepienia.zwierzę_id_zwierzę = wizyta.zwierzę_id_zwierzę) AS "Najnowsze Szczepienie"
FROM
    wizyta;
```

The results table has 10 rows and 5 columns: ID\_WIZYTA, DATA, GODZINA, Liczba Diagnoz, Średni Wiek Zwierzęcia, and Najnowsze Szczepienie.

ID_WIZYTA	DATA	GODZINA	Liczba Diagnoz	Średni Wiek Zwierzęcia	Najnowsze Szczepienie
1	115-JAN-23	01-JAN-24	1	610-JAN-23	
2	210-NOV-22	01-JAN-24	1	505-NOV-22	
3	320-MAR-23	01-JAN-24	1	715-MAR-23	
4	425-SEP-22	01-JAN-24	1	420-SEP-22	
5	515-MAY-23	01-JAN-24	1	812-MAY-23	
6	612-AUG-22	01-JAN-24	1	508-AUG-22	
7	728-FEB-23	01-JAN-24	1	628-FEB-23	
8	815-DEC-22	01-JAN-24	1	710-DEC-22	
9	925-APR-23	01-JAN-24	1	418-APR-23	
10	1030-OCT-22	01-JAN-24	1	825-OCT-22	

## 10. Analiza szczepień zwierząt z podziałem na poziom ochrony:

SELECT

z.id\_zwierzę,

z.imię AS "Zwierzę",

COUNT(ks.id\_karta\_szczepienia) AS "Liczba Szczepień",

MAX(ks.data\_ostatniego\_szczepienia) AS "Najnowsze Szczepienie",

CASE

WHEN COUNT(ks.id\_karta\_szczepienia) = 0 THEN 'Niezaszczepione'

WHEN COUNT(ks.id\_karta\_szczepienia) > 0 AND COUNT(ks.id\_karta\_szczepienia) < 3  
THEN 'Niski Poziom Ochrony'

WHEN COUNT(ks.id\_karta\_szczepienia) >= 3 AND COUNT(ks.id\_karta\_szczepienia) < 6  
THEN 'Średni Poziom Ochrony'

```

ELSE 'Wysoki Poziom Ochrony'

END AS "Stan Ochrony"

FROM

zwierzę z

LEFT JOIN

karta_szczeplenia ks ON z.id_zwierzę = ks.zwierzę_id_zwierzę

GROUP BY

z.id_zwierzę, z.imię;

```

Worksheet		Query Builder
1	SELECT	
2	z.id_zwierzę,	
3	z.imię AS "Zwierzę",	
4	COUNT(ks.id_karta_szczeplenia) AS "Liczba Szczepień",	
5	MAX(ks.data_ostatniego_szczeplenia) AS "Najnowsze Szczepienie",	
6	CASE	
7	WHEN COUNT(ks.id_karta_szczeplenia) = 0 THEN 'Niezaszczepione'	
8	WHEN COUNT(ks.id_karta_szczeplenia) > 0 AND COUNT(ks.id_karta_szczeplenia) < 3 THEN 'Niski Poziom Ochrony'	
9	WHEN COUNT(ks.id_karta_szczeplenia) >= 3 AND COUNT(ks.id_karta_szczeplenia) < 6 THEN 'Średni Poziom Ochrony'	
10	ELSE 'Wysoki Poziom Ochrony'	
11	END AS "Stan Ochrony"	
12	FROM	
13	zwierzę z	
14	LEFT JOIN	
15	karta_szczeplenia ks ON z.id_zwierzę = ks.zwierzę_id_zwierzę	
16	GROUP BY	
17	z.id_zwierzę, z.imię;	

Query Result x				
All Rows Fetched: 10 in 0.093 seconds				
ID_ZWIERZE	Zwierzę	Liczba Szczepień	Najnowsze Szczepienie	Stan Ochrony
1	1 Burek		110-JAN-23	Niski Poziom Ochrony
2	2 Luna		105-NOV-22	Niski Poziom Ochrony
3	3 Max		115-MAR-23	Niski Poziom Ochrony
4	4 Mia		120-SEP-22	Niski Poziom Ochrony
5	5 Rocky		112-MAY-23	Niski Poziom Ochrony
6	6 Daisy		108-AUG-22	Niski Poziom Ochrony
7	7 Charlie		128-FEB-23	Niski Poziom Ochrony
8	8 Zoe		110-DEC-22	Niski Poziom Ochrony
9	9 Oscar		118-APR-23	Niski Poziom Ochrony
10	10 Molly		125-OCT-22	Niski Poziom Ochrony

## 11. Znajdź lekarzy, którzy mają najwięcej wizyt w danym miesiącu z uwzględnieniem ich specjalizacji:

```

SELECT lekarz.id_lekarz, lekarz.imię, lekarz.nazwisko, specjalizacja.nazwa_specjalizacji,
COUNT(wizyta.id_wizyta) AS liczba_wizyt

FROM lekarz

JOIN specjalizacja ON lekarz.specjalizacja_id_specjalizacja = specjalizacja.id_specjalizacja

JOIN wizyta ON lekarz.id_lekarz = wizyta.lekarz_id_lekarz

WHERE EXTRACT(MONTH FROM wizyta.data_) = EXTRACT(MONTH FROM SYSDATE)

GROUP BY lekarz.id_lekarz, lekarz.imię, lekarz.nazwisko, specjalizacja.nazwa_specjalizacji

ORDER BY COUNT(wizyta.id_wizyta) DESC;

```

```

20 SELECT lekarz.id lekarz, lekarz.imię, lekarz.nazwisko, specjalizacja.nazwa_specjalizacji, COUNT(wizyta.id wizyta) AS liczba wizyt
21 FROM lekarz
22 JOIN specjalizacja ON lekarz.specjalizacja_id_specjalizacja = specjalizacja.id_specjalizacja
23 JOIN wizyta ON lekarz.id_lekarz = wizyta.lekarz_id_lekarz
24 WHERE EXTRACT(MONTH FROM wizyta.data_) = EXTRACT(MONTH FROM SYSDATE)
25 GROUP BY lekarz.id_lekarz, lekarz.imię, lekarz.nazwisko, specjalizacja.nazwa_specjalizacji
26 ORDER BY COUNT(wizyta.id wizyta) DESC;
27
28

```

Query Result x

SQL | All Rows Fetched: 1 in 0.174 seconds

ID_LEKARZ	IMIĘ	NAZWISKO	NAZWA_SPECJALIZACJI	LICZBA_WIZYT
1	Jan	Kowalski	Choroby psów i kotów	1

## 12. Znajdź leki, które zostały przepisane pacjentom w ostatnich 12 miesiącach:

SELECT lek.id\_lek, lek.nazwa\_leku, lek.choroba\_id\_choroba, wizyta.data\_

FROM lek

JOIN diagnoza ON lek.choroba\_id\_choroba = diagnoza.choroba\_id\_choroba

JOIN wizyta ON diagnoza.wizyta\_id\_wizyta = wizyta.id\_wizyta

WHERE wizyta.data\_ >= ADD\_MONTHS(SYSDATE, -12);

```

28 SELECT lek.id_lek, lek.nazwa_leku, lek.choroba_id_choroba, wizyta.data_
29 FROM lek
30 JOIN diagnoza ON lek.choroba_id_choroba = diagnoza.choroba_id_choroba
31 JOIN wizyta ON diagnoza.wizyta_id_wizyta = wizyta.id_wizyta
32 WHERE wizyta.data_ >= ADD_MONTHS(SYSDATE, -12);
33

```

Query Result x

SQL | All Rows Fetched: 5 in 0.038 seconds

ID_LEK	NAZWA_LEKU	CHOROBA_ID_CHOROBA	DATA_
1	1 Paracetamol		1 15-JAN-23
2	3 Antyhistaminik		3 20-MAR-23
3	5 Ventolin		5 15-MAY-23
4	7 Bentoks		7 28-FEB-23
5	9 Chemioterapia		9 25-APR-23

### 3.3. Algebra relacyjna

$$1. \pi_{\text{imię}, \text{data}_{\text{urodzenia}}, \text{płeć}, \text{umaszczzenie}, \text{nazwa}_{\text{gatunku}}, \text{imię} || ' ' || \text{nazwisko}} \left( \text{Zwierzę} \bowtie_{\text{gatunek}_{\text{id}_{\text{gatunek}}} = \text{id}_{\text{gatunekGatunek}}} \text{właściciel}_{\text{id}_{\text{właściciel}}} \right) \\ = \pi_{\text{id}_{\text{właścicielWłaściciel}}}$$

$$2. \sigma_{\text{EXTRACT}(\text{YEAR FROM SYSDATE}) - \text{EXTRACT}(\text{YEAR FROM data}_{\text{urodzenia}}) < 5} (\text{Zwierzę})$$

$$3. \pi_{\text{nazwa}_{\text{gatunku}}, \text{COUNT}(*)} \left( \text{Zwierzę} \bowtie_{\text{gatunek}_{\text{id}_{\text{gatunek}}} = \text{id}_{\text{gatunekGatunek}}} \right)$$

$$4. \sigma_{\text{id}_{\text{zwierzę}} \text{ IN } \left( \text{SELECT DISTINCT } \text{id}_{\text{zwierzę}} \text{ FROM wizyta WHERE data} > \text{SYSDATE} - \text{INTERVAL '10' MONTH} \right)} (\text{Zwierzę})$$

$$5. \pi_{\text{Rok}, \text{imię} || ' ' || \text{nazwisko}, \text{COUNT}(*)} (\text{Wizyta} \bowtie_{\text{lekarz}_{\text{id}_{\text{lekarz}}} = \text{id}_{\text{lekarzLekarz}}})$$

$$\text{gdzie Rok} = \text{EXTRACT}(\text{YEAR FROM data})$$

$$6. \pi_{\text{id}_{\text{zwierzę}}, \text{imię}, \text{data}}, \text{Numer Wizyty} \left( \rho_{\text{Numer Wizyty}} \left( \sigma_{\text{płeć} = 'Samica'} \left( \text{Zwierzę} \bowtie_{\text{id}_{\text{zwierzę}} = \text{id}_{\text{zwierzęWizyta}}} \right) \right) \right)$$

$$\text{gdzie Numer Wizyty} =$$

$$\text{ROW}_{\text{NUMBER OVER} \left( \text{PARTITION BY } \text{id}_{\text{zwierzę}} \text{ ORDER BY data}_{\text{DESC}} \right)}$$

$$7. \pi_{\text{id}_{\text{wizyta}}, \text{data}, \text{godzina}, \text{Typ Dnia}}$$

$$\left( \sigma_{\text{TO}_{\text{CHAR}(\text{data}, 'YYYY-MM')} \text{ IN } \left( \text{SELECT TO}_{\text{CHAR}(\text{data}, 'YYYY-MM')} \text{ FROM Wizyta GROUP BY TO}_{\text{CHAR}(\text{data}, 'YYYY-MM')} \text{ HAVING COUNT(DISTINCT lekarz}_{\text{id}_{\text{lekarz}}}) = 1 \right)} \right)$$

$$8. \pi_{\text{id}_{\text{zwierzę}}, \text{imię}, \text{data}_{\text{urodzenia}}, \text{Kategoria}_{\text{Wiekowa}}(\sigma_{\text{data}_{\text{urodzenia}} \neq \text{NULL}}(\text{Zwierzę}))}$$

## 4. Określenie kierunków rozwoju aplikacji

### 1. Usprawnienie interfejsu graficznego:

Dodanie intuicyjnego interfejsu graficznego dla personelu kliniki i właścicieli zwierząt, co ułatwi korzystanie z aplikacji.

### 2. Rozszerzenie funkcji analizy danych:

Dodanie zaawansowanych narzędzi analizy danych i generowania raportów, umożliwiających lepsze zrozumienie statystyk dotyczących wizyt, diagnoz, leków itp.

Implementacja systemu powiadomień i alarmów, informujących o ważnych wydarzeniach, takich jak terminy szczepień czy zalecone badania.

### 3. Integracja z systemami zewnętrznymi:

Stworzenie interfejsów do integracji z innymi systemami medycznymi czy laboratoriami, umożliwiając płynny przepływ informacji między różnymi instytucjami.

Wprowadzenie możliwości przesyłania danych do systemów rozliczeniowych lub ubezpieczycieli.

### 4. Wsparcie telemedycyny:

Ewentualne wprowadzenie funkcji telemedycznych umożliwiających konsultacje zdalne z lekarzami weterynarii.

Integracja z platformami do zdalnej diagnozy i porad.

### 5. Dalsza edukacja i szkolenia:

Udostępnienie materiałów edukacyjnych i szkoleń online dla personelu klinicznego, aby pomóc im w pełnym wykorzystaniu potencjału aplikacji.

Organizacja webinarów czy szkoleń dotyczących nowych funkcji i aktualizacji.

### 4.1. Literatura:

- "SQL Performance Explained" autorstwa Markus Winand
- <https://www.w3schools.com/sql/>
- <https://stackoverflow.com>

## 5. Podsumowanie

Projekt obejmuje stworzenie kompleksowej bazy danych dla kliniki weterynaryjnej, umożliwiającej zarządzanie informacjami dotyczącymi pacjentów, wizyt, lekarzy, leków, szczepień, właścicieli zwierząt i wielu innych istotnych danych. Początkowo zdefiniowane zostały tabele obejmujące informacje o adresach, chorobach, diagnozach, gabinetach, gatunkach zwierząt, kartach szczepień, lekach, lekarzach, rodzajach zwierząt, specjalizacjach, wizytach, właścicielach, zabiegach i samych zwierzętach. Przykłady obejmują relacje między diagnozami a chorobami, kartami szczepień a zwierzętami, lekami a chorobami, czy też wizytami a lekarzami. Dzięki temu baza danych jest spójna i umożliwia efektywne przetwarzanie informacji.

Dodatkowo, projekt uwzględnia dodanie przykładowych danych do poszczególnych tabel poprzez klauzulę INSERT, co pozwala na lepsze zrozumienie funkcjonalności bazy. Przykładowe zapytania SQL zostały stworzone w celu pokazania możliwości analizy danych, takie jak wyświetlanie informacji o zwierzętach, selekcja zwierząt według kryteriów wiekowych, ilość zwierząt według gatunków czy statystyki dotyczące wizyt lekarskich.

Podsumowując, baza danych została zaprojektowana tak, aby umożliwiała kompleksowe zarządzanie danymi związanymi z kliniką weterynaryjną. Warunki rozwoju aplikacji obejmują dalszą optymalizację struktury bazy danych, implementację dodatkowych funkcji do obsługi specyficznych przypadków klinicznych, oraz ewentualne rozbudowy o interfejs graficzny ułatwiający korzystanie z systemu przez personel kliniki.