МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИИ
«Санкт-Петербургский Политехнический университет Петра Великого»
Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта

Курс: Теоретические основы баз данных

Отчет по курсовой работе «База данных ветеринарной клиники»

Выполнила Андреева Наталия Сергеевна
студентка группы 3530201/90101

Принял Попов Сергей Геннадьевич

Санкт-Петербург, 2022

Содержание

1 Аналитика

1.1 Описание предметной области

Ветеринарная клиника — это специализированное ветеринарное учреждение, которое оказывает ветеринарные услуги диагностического, лечебного и профилактического характера в специально приспособленном и оборудованном здании и помещениях.

Пациентами ветеринарной клиники являются животные различных видов: кошки, собаки, хомяки, попугаи и др. На каждого пациента заводится ветеринарная карта, в которую заносится вид животного, его кличка, дата рождения и пол.

Клиентами ветеринарной клиники являются хозяева пациентов. В ветеринарной карте, кроме информации о пациентах, присутствует информация о клиентах: фамилия, имя, отчество, телефон.

Ветклиника предоставляет различные медицинские услуги. Среди них терапия; хирургическое вмешательство; исследования, в числе которых проведение анализов, УЗИ, ЭКГ, рентген; консультирование; эвтаназия и др. Каждую из услуг может провести врач определенной специальности. Но одну и ту же услугу могут предоставить врачи разных специальностей. С врачем можно связаться по телефону в рабочие часы.

Также возможно пребывание пациента в стационаре. Стационар — это подразделение ветеринарной клиники с постоянным присутствием в нем пациента. Стационар необходим пациентам, перенесшим операции, получившим травмы, а также в других ситуациях, при которых необходим постоянный мониторинг состояния пациента. Стационар представляет собой помещение с индивидуальными клетками для пациентов. Различные показатели (температура, артериальное давление, аппетит, симптомы) каждого пациента фиксируются с определенной периодичностью, которая зависит от степени тяжести состояния пациента. В стационаре предоставляются различные услуги, например, выдача лекарств, проведение инъекций, кормление и др.

В ветеринарную карту заносится дата поступления в стационар, дата выписки, причина поступления в стационар и все услуги, которые проводились во время пребывания пациента в стационаре, а также информация о врачах, следивших за состоянием пациента и осуществлявших медицинские услуги. Клиент также должен оплатить все оказанные в стационаре услуги.

Кроме стационара, медицинские услуги также проводятся во время приема. Когда домашнему животному клиента требуется медицинская помощь, клиент либо записывается на прием по телефону ветклиники, либо приходит на прием в порядке живой очереди. Если пациент находится в экстренном состоянии, то его принимают без очереди. Прием проходит в определенные дату и время. Вся информация по приему, а именно: лечащий врач, предоставленные услуги, поставленные диагнозы, назначение врача — заносится в ветеринарную карту. По окончании приема, клиент обязан оплатить все оказанные услуги.

Помимо предоставления медицинских услуг ветеринарная клиника занимается продажей товаров для животных. Каждый товар имеет свою группу, например, лекарство, корм и пр. Ведется учет продаж товаров. Клиент совершает покупку одного

или нескольких товаров по установленной цене. Учет продаж предполагает указание даты продажи каждого товара, а также информации о клиенте.

1.2 Функции системы

Выделены следующие функции системы:

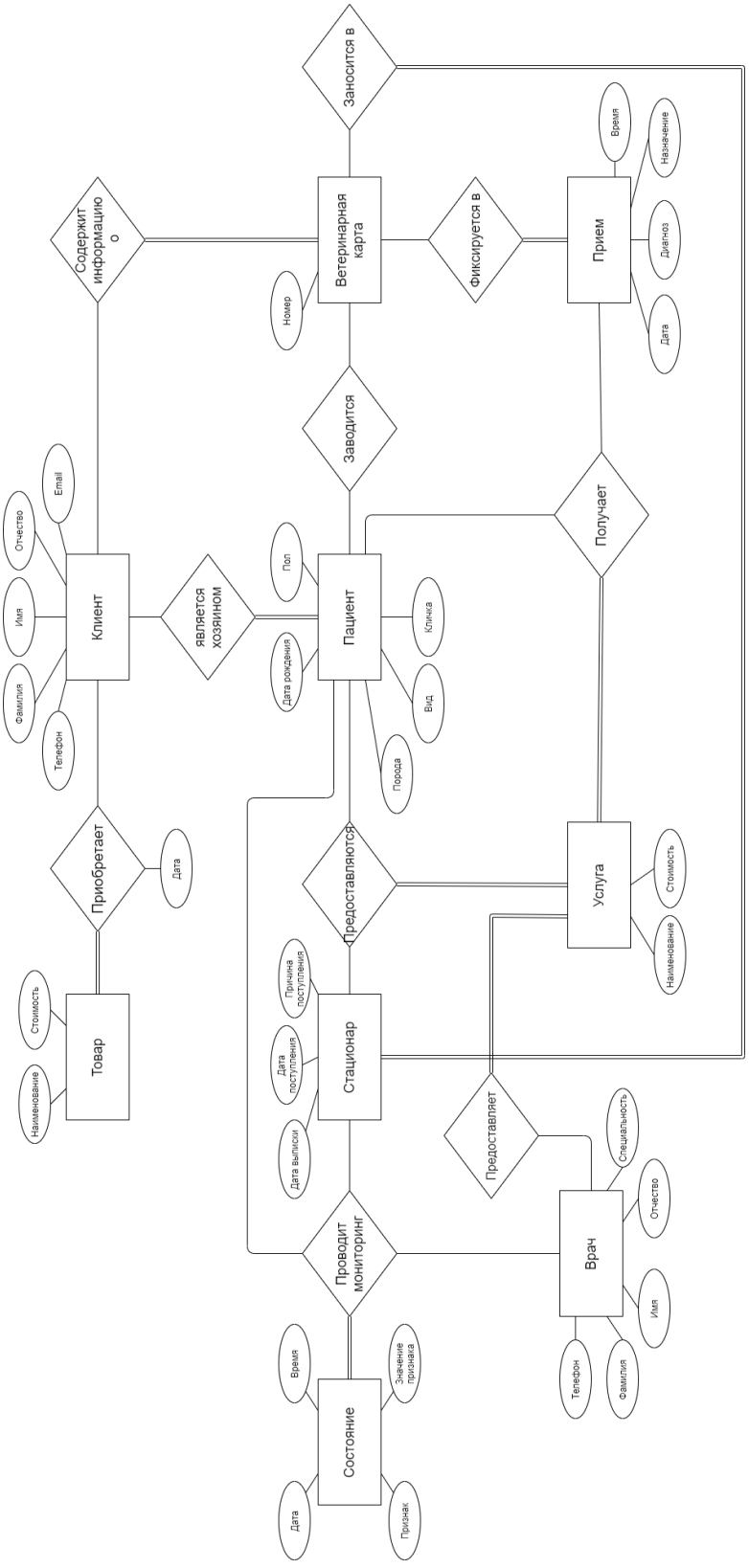
- учет клиентов и пациентов ветклиники;
- учет услуг, предоставленных пациентам;
- учет продаж товаров.

1.3 Выделение сущностей и их атрибутов

- Клиент:
  - фамилия;
  - имя;
  - отчество;
  - телефон;
- Пациент:
  - кличка;
  - вид;
  - дата рождения;
  - пол;
- Врач:
  - фамилия;
  - имя;
  - отчество;
  - специальность;
  - телефон;
- Ветеринарная карта:
  - номер;
- Прием:
  - дата;
  - время;

  - диагноз;
  - назначение;
- Стационар:
  - дата поступления;
  - дата выписки;
  - причина поступления в стационар;
- Состояние:
  - дата фиксации состояния;
  - время фиксации состояния;
  - фиксируемый признак;
  - значение признака;
- Услуга:
  - наименование услуги;
  - стоимость;
- Товар:
  - наименование товара;
  - стоимость;

1.4 ER-диаграмма



1.5 Чтение ER-диаграммы

- Клиент является хозяином пациента.
- На пациента заводится ветеринарная карта.
- Ветеринарная карта содержит информацию о клиенте.
- Пациент получает услуги на приеме.
- Пациенту проводятся услуги в стационаре.
- Прием фиксируется в ветеринарной карте.
- Нахождение в стационаре заносится в ветеринарную карту.
- В стационаре проводится мониторинг состояния пациента.
- Клиент приобретает товары в определенную дату.
- Врач проводит мониторинг состояния пациента, находящегося в стационаре.
- Врач предоставляет услуги.

1.6 Схема объектов

Рис. 1: Схема объектов

2 Проектирование

2.1 Схема базы данных

На рисунке 2 изображена схема базы данных ветклиники на русском языке. На рисунке 3 изображена схема базы данных ветклиники на английском языке.

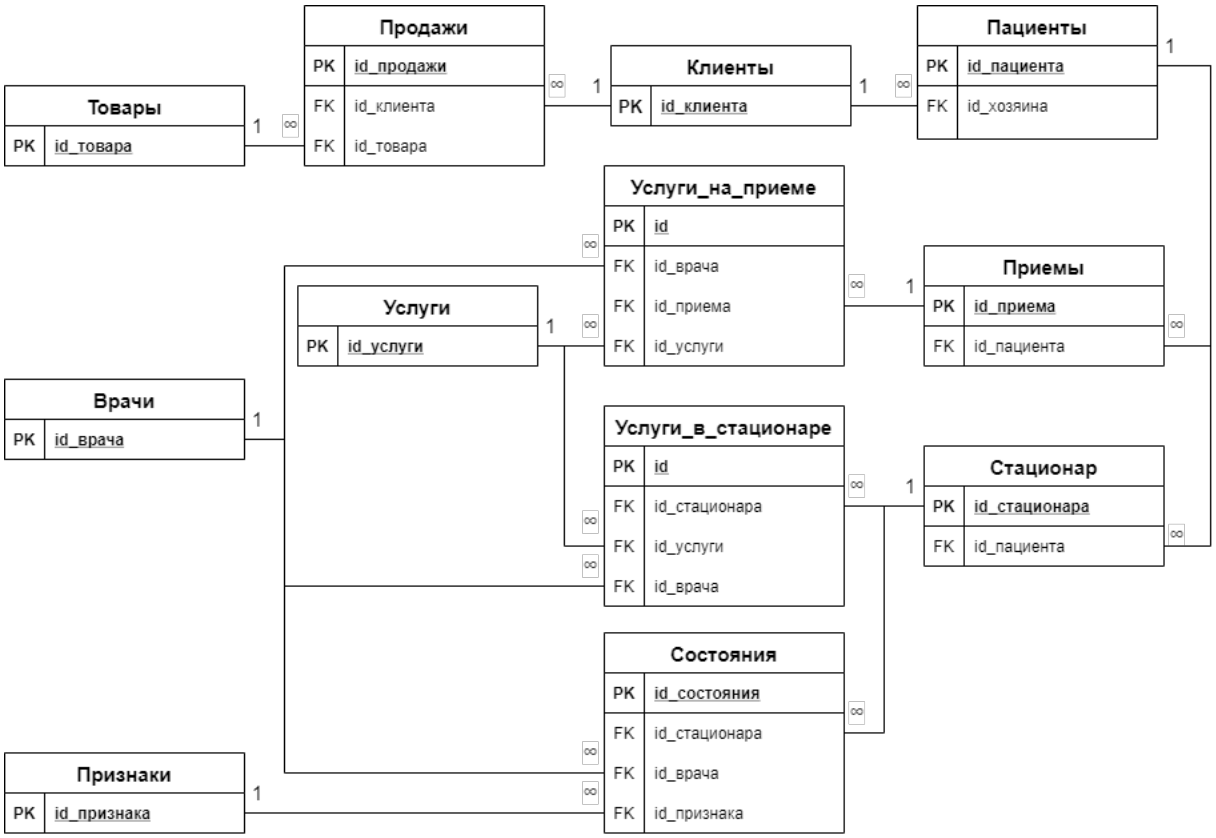


Рис. 2: Схема базы данных на русском языке

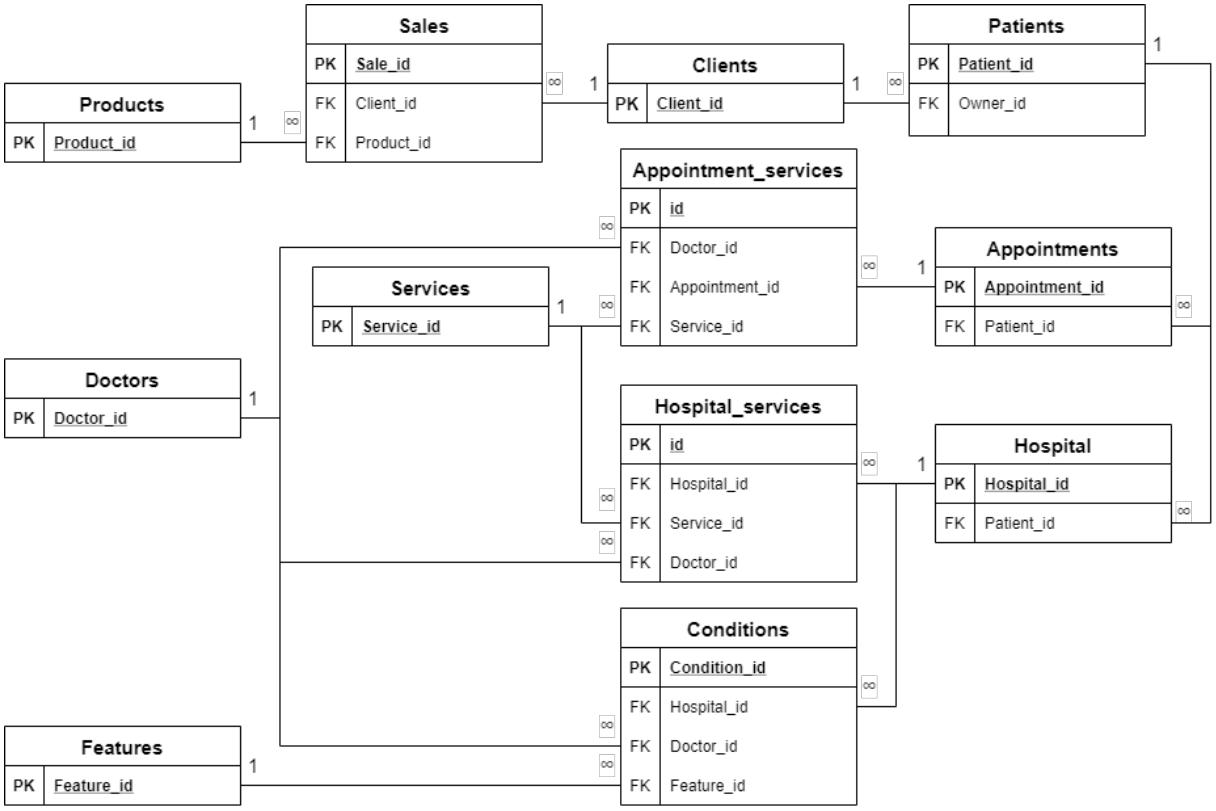


Рис. 3: Схема базы данных на английском языке

2.2 Таблицы базы данных

В таблицах 1-12 указаны атрибуты таблиц проектируемой базы данных и типы данных поля. Если поле является ключом, то указан его тип: PK – первичный ключ, FK – внешний ключ.

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_клиента	Client_id	INT	PK	-
Фамилия	Second_name	VARCHAR(30)	-	-
Имя	First_name	VARCHAR(30)	-	-
Отчество	Patronymic	VARCHAR(35)	-	-
Номер_телефона	Phone_number	VARCHAR(20)	-	-

Таблица 1: Клиенты (Clients)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_товара	Product_id	INT	PK	-
Наименование_товара	Product_name	VARCHAR(30)	-	-
Стоимость	Price	DECIMAL(6,2)	-	-

Таблица 2: Товары (Products)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_продажи	Sale_id	INT	PK	-
id_клиента	Client_id	INT	FK	Clients: Client_id
id_товара	Product_id	INT	FK	Products: Product_id
Дата_продажи	Sale_date	DATE	-	-

Таблица 3: Продажи (Sales)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_пациента	Patient_id	INT	PK	-
id_хозяина	Owner_id	INT	FK	Clients: Client_id
Кличка	Moniker	VARCHAR(30)	-	-
Дата_рождения	Birthday	DATE	-	-
Вид_животного	Animal_kind	VARCHAR(20)	-	-
Пол	Gender	CHAR(1)	-	-

Таблица 4: Пациенты (Patients)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_врача	Doctor_id	INT	PK	-
Фамилия	Second_name	VARCHAR(30)	-	-
Имя	First_name	VARCHAR(30)	-	-
Отчество	Patronymic	VARCHAR(35)	-	-
Номер_телефона	Phone_number	VARCHAR(20)	-	-

Таблица 5: Врачи (Doctors)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_приема	Appointment_id	INT	PK	-
id_пациента	Patient_id	INT	FK	Patients: Patient_id
Дата_приема	Appointment_date	DATE	-	-
Время_приема	Appointment_time	TIME	-	-

Таблица 6: Приемы (Appointments)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_услуги	Service_id	INT	PK	-
Наименование_услуги	Service_name	VARCHAR(50)	-	-
Стоимость	Price	DECIMAL(6,2)	-	-

Таблица 7: Услуги (Services)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id	id	INT	PK	-
id_врача	Doctor_id	INT	FK	Doctors: Doctor_id
id_приема	Appointment_id	INT	FK	Appointment: Appointment_id
id_услуги	Service_id	INT	FK	Services: Service_id

Таблица 8: Услуги на приеме (Appointment services)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_стационара	Hospital_id	INT	PK	-
id_пациента	Patient_id	INT	FK	Patients: Patient_id
Дата_поступления	Admission_date	DATE	-	-
Дата_выписки	Discharge_date	DATE	-	-

Таблица 9: Стационар (Hospital)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id	id	INT	PK	-
id_врача	Doctor_id	INT	FK	Doctors: Doctor_id
id_стационара	Hospital_id	INT	FK	Hospital: Hospital_id
id_услуги	Service_id	INT	FK	Services: Service_id

Таблица 10: Услуги в стационаре (Hospital services)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_признака	Feature_id	INT	PK	-
Название_признака	Feature_name	VARCHAR(50)	-	-

Таблица 11: Признаки (Features)

Название атрибута (рус.)	Название атрибута (англ.)	Тип	Тип ключа	Ссылка
id_состояния	Condition_id	INT	PK	-
id_врача	Doctor_id	INT	FK	Doctors: Doctor_id
id_стационара	Hospital_id	INT	FK	Hospital: Hospital_id
id_признака	Feature_id	INT	FK	Features: Feature_id
Значение_признака	Feature_value	VARCHAR(50)	-	-
Дата_фиксации	Fixation_date	DATE	-	-
Время_фиксации	Fixation_time	TIME	-	-

Таблица 12: Состояния (Conditions)

3 Программирование

3.1 Создание базы данных

В графическом клиенте MySQL Workbench бал написан скрипт для генерации спроектированной базы данных. Текст скрипта приведен в приложении А.
Процесс создания базы данных:

- с помощью команды CREATE DATABASE создается пустая база данных VetClinic;
- с помощью команды CREATE TABLE создаются таблицы, описанные в пункте 2.2;
- с помощью AUTO_INCREMENT для первичных ключей установлено автоматическое увеличение значения столбца при добавлении записи.

3.2 Заполнение базы данных

Для заполнения базы данных была написана программа на языке Java. Код программы приведен в приложении Б.
Программа содержит 2 класса. Класс Generator предоставляет методы для генерации строк таблиц. Класс Filler осуществляет подключение к базе данных и заполняет таблицы с помощью методов класса Generator.
В таблице 13 приведено количество записей в каждой из таблиц, а также порядок заполнения таблиц. Всего в таблицах 202 890 записей.

Название таблицы	Порядок заполнения	Количество записей
Clients	1	500
Products	1	50
Doctors	1	20
Services	1	50
Features	1	20
Patients	2	1 500
Sales	2	2750
Appointments	3	11 250
Hospital	3	6 750
Appointment_services	4	61 875
Hospital_services	4	67 500
Conditions	4	50 625

Таблица 13: Количество записей и порядок заполнения таблиц

4 Запросы к базе данных

4.1 Запрос №1а

Формулировка запроса: вывести всех пациентов, которым продавали товар A и которых принимал врач B.
Текст запроса:

```
SELECT Moniker, Animal_kind, Birthday
FROM ((Sales JOIN Patients
ON (Product_id=19 AND Client_id=Owner_id)) JOIN
Appointments USING (Patient_id)) JOIN Appointment_services
ON (Doctor_id = 18 AND
Appointments.Appointment_id=Appointment_services.Appointment_id)
GROUP BY Patient_id;
```

На рисунке 4 приведен результат выполнения запроса. Найдено 145 строк, время выполнения — 0.0197 с.

Moniker	Animal_kind	Birthday
Арчи	попугай	2006-08-30
Хэнк	кошка	2020-05-29
Буян	кролик	2016-07-17
Фея	попугай	2011-01-17
Симона	попугай	2015-06-17
Оливия	попугай	2004-09-21
Вилли	хомяк	2002-08-16
Бруно	кошка	2008-02-18
Микки	хомяк	2000-06-24
Трикси	попугай	2019-07-10
Микки	попугай	2001-10-30
Трикси	кролик	2008-09-11
Лука	кролик	2010-07-03
Лили	хомяк	2014-09-04
Честер	попугай	2004-08-20
Арчи	собака	2007-03-08
Дина	кошка	2008-04-19
Фокси	хомяк	2014-03-13
Дюк	хомяк	2011-04-06
Стелла	хомяк	2015-11-15
Фиона	хомяк	2019-04-10
Ник	кролик	2005-06-08
Ава	кошка	2012-05-15
Бони	попугай	2018-11-08
Ночка	хомяк	2019-06-06
Маркиза	собака	2002-09-09

Рис. 4: Результат выполнения запроса №1а

EXPLAIN запроса:

id	select_type	table	paritions	type	possible_keys	key
1	SIMPLE	Sales	NULL	ref	Client_id, Product_id	Product_id
1	SIMPLE	Patients	NULL	ref	PRIMARY,Owner_id	Owner_id
1	SIMPLE	Appointments	NULL	ref	PRIMARY,Patient_id	Patient_id
1	SIMPLE	Appointment_services	NULL	ref	Doctor_id,Appointment_id	Appointment_id

key_len	ref	rows	filtered	Extra
5	const	53	100.00	Using where; Using temporary
5	vetclinic.Sales.Client_id	3	100.00	NULL
5	vetclinic.Patients.Patient_id	7	100.00	Using index
5	vetclinic.Appointments.Appointment_id	5	5.06	Using where

Таблица 14: EXPLAIN для запроса №1a

Объяснение запроса: выполняется 4 независимых запроса типа SIMPLE, т.е. без подзапросов и UNION'ов. Из таблицы продаж (Sales) выбираются строки с товаром A (в данном случае товар с id 19) и происходит связывание с таблицей пациентов (Patients). Также происходит связывание с таблицей приемов (Appointments) по Patient_id. В таблице услуг на приеме (Appointment_services) выбираются строки с врачом B (в данном случае врач с id 18) и производится связывание по колонке Appointment_id.

4.2 Запрос №1b

Формулировка запроса: вывести всех пациентов, которым продавали товар A не менее 3 раз и которых принимал врач B.
Текст запроса:

```
SELECT DISTINCT Moniker, Animal_kind, Birthday, Sales_number
FROM (SELECT Client_id, count(*) AS Sales_number
FROM Sales WHERE Product_id = 19
GROUP BY Client_id HAVING Sales_number >= 3) AS Number_of_sales JOIN
Patients ON Client_id=Owner_id JOIN
Appointments USING (Patient_id) JOIN
Appointment_services USING (Appointment_id)
WHERE Doctor_id = 18;
```

На рисунке 5 изображен результат выполнения запроса. Выбрано 5 записей, время выполнения — 0.0020 с.

```
+---------+-------------+------------+--------------+
| Moniker | Animal_kind | Birthday   | Sales_number |
+---------+-------------+------------+--------------+
| Симона  | собака      | 2000-04-24 |            3 |
| Дилли   | хомяк       | 2014-11-02 |            3 |
| Ава     | кролик      | 2005-10-03 |            3 |
| Бони    | собака      | 2000-08-25 |            3 |
| Тесса   | кролик      | 2007-01-07 |            3 |
+---------+-------------+------------+--------------+
5 rows in set (0.0020 sec)
```

Рис. 5: Результат выполнения запроса №1b

EXPLAIN запроса:

id	select_type	table	paritions	type	possible_keys	key
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL
1	PRIMARY	Patients	NULL	ref	PRIMARY,Owner_id	Owner_id
1	PRIMARY	Appointments	NULL	ref	PRIMARY,Patient_id	Patient_id
1	PRIMARY	Appointment_services	NULL	ref	Doctor_id,Appointment_id	Appointment_id
2	DERIVED	Sales	NULL	ref	Client_id,Product_id	Product_id

key_len	ref	rows	filtered	Extra
NULL	NULL	53	100.00	Using where; Using temporary
5	Number_of_sales.Client_id	3	100.00	NULL
5	vetclinic.Patients.Patient_id	7	100.00	Using index
5	vetclinic.Appointments.Appointment_id	5	5.06	Using where
5	const	53	100.00	Using temporary

Таблица 15: EXPLAIN для запроса №1b

Объяснение запроса: запрос типа DERIVED выбирает клиентов, которые покупали товар A не менее 3 раз. Полученная таблица связывается с таблицей пациентов (Patients), приемов (Appointments) и услуг на приеме (Appointment_services).

4.3 Запрос №2

Формулировка запроса: посчитать количество пациентов, которых принимал врач A и на приеме, и в стационаре.
Текст запроса:

```
SELECT Second_name, First_name, Patronymic, count(*) AS Patient_number
FROM ((SELECT Doctor_id, Patient_id
FROM Appointment_services JOIN Appointments
ON (Doctor_id = 10 AND
Appointment_services.Appointment_id = Appointments.Appointment_id)
GROUP BY Patient_id) AS From_appointments
JOIN (SELECT Doctor_id, Patient_id
FROM Hospital_services JOIN Hospital
```

```
ON (Doctor_id = 10 AND Hospital_services.Hospital_id=Hospital.Hospital_id)
GROUP BY Patient_id) AS From_hospital
USING (Doctor_id, Patient_id)) JOIN Doctors
USING (Doctor_id);
```

На рисунке 6 изображен результат выполнения запроса. Выбрана 1 запись, время выполнения — 0.0670 с.

```
+-------------+------------+------------+----------------+
| Second_name | First_name | Patronymic | Patient_number |
+-------------+------------+------------+----------------+
| Семенова    | Светлана   | Витальевна |            896 |
+-------------+------------+------------+----------------+
1 row in set (0.0670 sec)
```

Рис. 6: Результат выполнения запроса №2

EXPLAIN запроса:

id	select_type	table	type	possible_keys	key	key_len
1	PRIMARY	Doctors	ALL	PRIMARY,	NULL	NULL
1	PRIMARY	<derived3>	ref	<auto_key0>	<auto_key0>	5
1	PRIMARY	<derived2>	ref	<auto_key0>	<auto_key0>	10
3	DERIVED	Hospital_services	ref	Doctor_id,Hospital_id	Doctor_id	5
3	DERIVED	Hospital	eq_ref	PRIMARY,Patient_id	PRIMARY	4
2	DERIVED	Appointment_services	ref	Appointment_id,Doctor_id	Doctor_id	5
2	DERIVED	Appointments	eq_ref	PRIMARY,Patient_id	PRIMARY	4

ref	rows	filtered	Extra
NULL	20	100.00	NULL
vetclinic.Doctors.Doctor_id	35	100.00	Using where; Using index
vetclinic.Doctors.Doctor_id,From_hospital.Patient_id	31	100.00	Using index
const	3466	100.00	Using where; Using temporary
vetclinic.Hospital_services.Hospital_id	1	100.00	NULL
const	3109	100.00	Using where; Using temporary
vetclinic.Appointment_services.Appointment_id	1	100.00	NULL

Таблица 16: EXPLAIN для запроса №2

Объяснение запроса: вложенные запросы формируют таблицы врача A (в данном случае с id 10) и пациентов, которых принимал этот врач на приеме (From_appointments) и в стационаре (From_hospital). Из созданных таблиц формируется таблица такая, что в ней присутствуют строки, которые присутствуют в обоих таблицах. Таблица связывается с таблицей врачей (Doctors).

4.4 Запрос №3

Формулировка запроса: для каждого врача посчитать число услуг на приеме и число услуг в стационаре. Построить гистограмму.

Текст запроса:

```
SELECT Second_name,
Appointment_services_number,
Hospital_services_number
FROM ((SELECT count(*) AS Appointment_services_number, Doctor_id
FROM Appointment_services
GROUP BY Doctor_id) AS Services_on_appointments JOIN
(SELECT count(*) AS Hospital_services_number, Doctor_id
FROM Hospital_services
GROUP BY Doctor_id) AS Services_in_hospital
USING (Doctor_id)) JOIN Doctors
USING (Doctor_id);
```

На рисунке 7 изображен результат выполнения запроса. Выбрано 20 строк, время выполнения — 0.0433 с.

Second_name	Appointment_services_number	Hospital_services_number
Макаров	3089	3400
Волков	2975	3376
Шишокин	3105	3436
Шарикова	3110	3294
Табаков	3104	3363
Новиков	3100	3328
Тараканов	3035	3269
Киселева	3206	3321
Петров	3119	3528
Семенова	3109	3466
Полякова	3081	3331
Захаров	3146	3347
Залежнева	3097	3327
Бородина	3068	3397
Алексеева	3007	3391
Романова	3080	3395
Залежнев	3147	3403
Иванова	3156	3388
Закатов	3044	3418
Хворосткова	3097	3322

20 rows in set (0.0433 sec)

Рис. 7: Результат выполнения запроса №3

EXPLAIN запроса:

id	select_type	table	paritions	type	possible_keys	key
1	PRIMARY	Doctors	NULL	ALL	PRIMARY	NULL
1	PRIMARY	<derived2>	NULL	ref	<auto_key0>	<auto_key0>
1	PRIMARY	<derived3>	NULL	ref	<auto_key0>	<auto_key0>
3	DERIVED	Hospital_services	NULL	index	Doctor_id	Doctor_id
2	DERIVED	Appointment_services	NULL	index	Doctor_id	Doctor_id

key_len	ref	rows	filtered	Extra
NULL	NULL	20	100.00	NULL
5	vetclinic.Doctors.Doctor_id	623	100.00	NULL
5	vetclinic.Doctors.Doctor_id	678	100.00	NULL
5	NULL	67893	100.00	Using index
5	NULL	62342	100.00	Using index

Таблица 17: EXPLAIN для запроса №3

Объяснение запроса: вложенные запросы формируют таблицы врачей с количеством услуг на приеме (Services_on_appointments) и в стационаре (Services_in_hospital). Эти таблицы связываются по Doctor_id. Таким же образом происходит соединение с таблицей врачей (Doctors).
На рисунке 8 приведена гистограмма, построенная по результатам выполнения запроса.

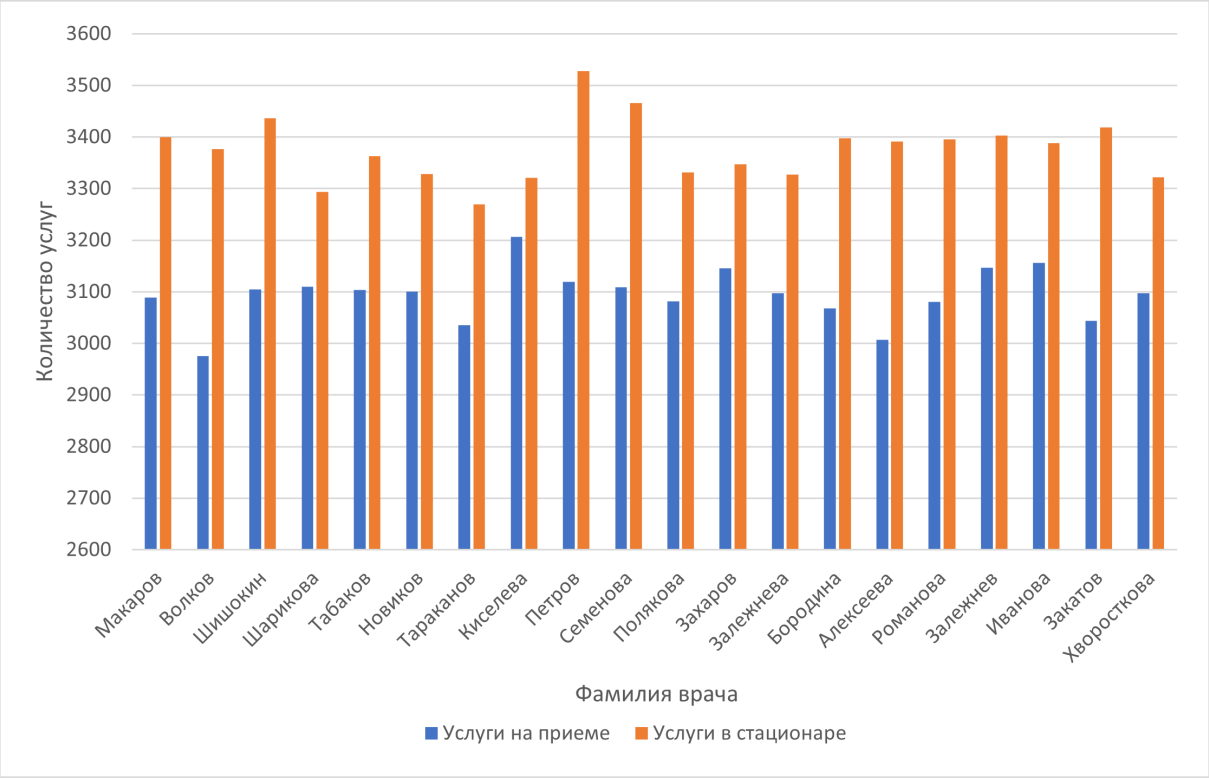


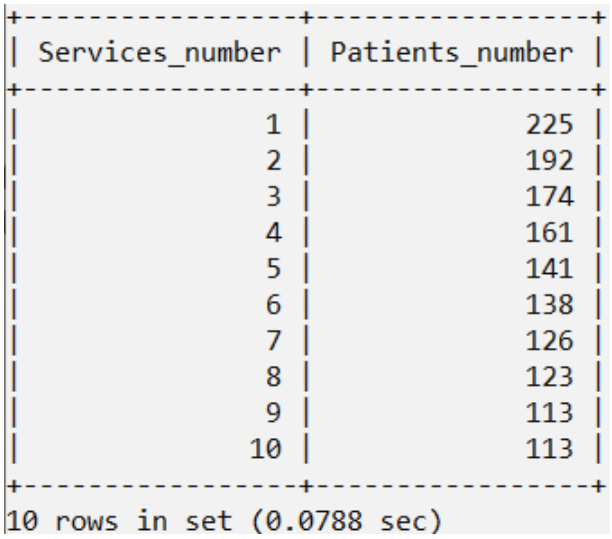
Рис. 8: Гистограмма к запросу №3

4.5 Запрос №4

Формулировка запроса: посчитать число пациентов с одинаковым числом услуг на приеме. Построить гистограмму.
Текст запроса:

```
SELECT Services_number, count(*) AS Patients_number
FROM (SELECT DISTINCT Patient_id, count(*) AS Services_number
FROM Appointment_services JOIN Appointments
USING (Appointment_id)
GROUP BY Appointment_id) AS Number_of_services
GROUP BY Services_number;
```

На рисунке 9 представлен результат выполнения запроса. Выбрано 10 записей, время выполнения — 0.0788 с.



Services_number	Patients_number
1	225
2	192
3	174
4	161
5	141
6	138
7	126
8	123
9	113
10	113

10 rows in set (0.0788 sec)

Рис. 9: Результат выполнения запроса №4

EXPLAIN запроса:

id	select_type	table	paritions	type	possible_keys	key
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL
2	DERIVED	Appointments	NULL	index	PRIMARY	Patient_id
2	DERIVED	Appointment_services	NULL	ref	Appointment_id	Appointment_id

key_len	ref	rows	filtered	Extra
NULL	NULL	61133	100.00	Using temporary
5	NULL	11482	100.00	Using index; Using temporary
5	vetclinic.Appointments.Appointment_id	5	100.00	Using index

Таблица 18: EXPLAIN для запроса №4

Объяснение запроса: во вложенном запросе по каждому пациенту происходит подсчет количества услуг на приеме. Во внешнем запросе для каждого числа услуг производится подсчет пацеиентов.
На рисунке 10 приведена гистограмма, построенная по результатам выполнения запроса.

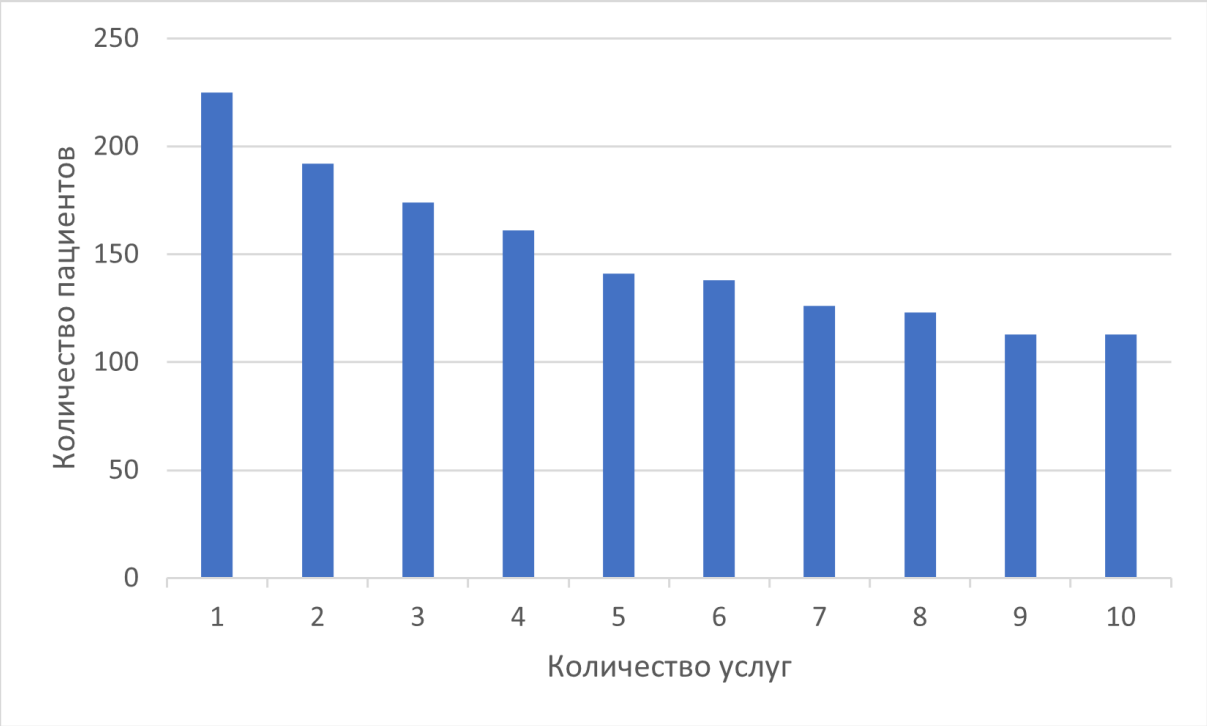


Рис. 10: Гистограмма к запросу №4

4.6 Запрос №5

Формулировка запроса: найти клиентов с максимальным числом продаж.
Текст запроса:

```
SELECT Second_name, First_name, Patronymic, count(*) AS Sales_number
FROM Sales JOIN Clients USING (Client_id)
GROUP BY Client_id
HAVING Sales_number =
(SELECT count(*) AS Number_of_sales
FROM Sales
GROUP BY Client_id
ORDER BY Number_of_sales DESC
LIMIT 1);
```

На рисунке 11 изображен результат выполнения запроса. Выбрано 50 записей, время выполнения — 0.0147 с.

Second_name	First_name	Patronymic	Sales_number
Яковлева	Вероника	Петровна	10
Шариков	Виталий	Евгеньевич	10
Гончаров	Никита	Артемович	10
Соколова	Ольга	Евгеньевна	10
Сорокина	Светлана	Кирилловна	10
Виноградов	Владимир	Андреевич	10
Михайлова	Вера	Алексеевна	10
Сидорова	Оксана	Валерьевна	10
Кузнецова	Людмила	Сергеевна	10
Соколов	Евгений	Николаевич	10
Бородин	Петр	Петрович	10
Варенников	Георгий	Глебович	10
Петрова	Александра	Дмитриевна	10
Волков	Максим	Львович	10
Захаров	Виктор	Кириллович	10
Хаборов	Семен	Васильевич	10
Лебедева	Елена	Александровна	10
Николаева	Анастасия	Егоровна	10
Лапин	Евгений	Григоревич	10
Петров	Иван	Анатольевич	10
Михайлов	Андрей	Владирович	10
Павлов	Илья	Егорович	10
Вареникова	Вероника	Алексеевна	10
Новикова	Анастасия	Семеновна	10
Макаров	Сергей	Анатольевич	10
Яковлев	Артем	Сергеевич	10
Хворосткова	Любовь	Витальевна	10
Варенников	Михаил	Александрович	10
Морозов	Василий	Евгеньевич	10
Захаров	Василий	Валерьевич	10
Макаров	Виталий	Анатольевич	10
Червяков	Владимир	Владирович	10
Сорокин	Анатолий	Алексеевич	10
Хаборов	Дмитрий	Викторович	10
Егоров	Алексей	Викторович	10
Лапина	Анна	Егоровна	10
Решетов	Василий	Егорович	10
Васильева	Дарья	Михайловна	10

Рис. 11: Результат выполнения запроса №5

EXPLAIN запроса:

id	select_type	table	paritions	type	possible_keys	key	key_len
1	PRIMARY	Clients	NULL	ALL	PRIMARY	NULL	NULL
1	PRIMARY	Sales	NULL	ref	Client_id	Client_id	5
2	SUBQUERY	Sales	NULL	index	Client_id	Client_id	5

ref	rows	filtered	Extra
NULL	500	100.00	Using temporary
vetclinic.Clients.Client_id	5	100.00	Using index
NULL	2750	100.00	Using index; Using temporary; Using filesort

Таблица 19: EXPLAIN для запроса №5

Объяснение запроса: подзапрос находит максимальное число продаж, внешний запрос подсчитывает количество продаж для каждого клиента и оставляет записи с количеством продаж, совпадающими с результатом подзапроса.

4.7 Запрос №6

Формулировка запроса: для каждого врача посчитать число услуг на приеме по каждой услуге. Построить 3D-гистограмму.
Текст запроса:

```
SELECT Second_name, Service_name, count(*) AS Service_number
FROM Appointment_services JOIN Doctors
USING (Doctor_id) JOIN Services
USING (Service_id)
GROUP BY Doctor_id, Service_id;
```

На рисунке 12 представлен результат выполнения запроса. Выбрано 1000 записей, время выполнения — 0.1981 с.

Second_name	Service_name	Service_number
Макаров	Резекция опухоли на коже	65
Макаров	Проведение КТ	72
Макаров	Офтальмоскопия	69
Макаров	Иссечение части желудка	61
Макаров	Удаление папилломы	54
Макаров	Ампутация конечности	68
Макаров	Санирование ротовой полости	70
Макаров	Лечение органов зрения	70
Макаров	Удаление репродуктивных органов	55
Макаров	Лечение переломов челюстей	55
Макаров	Вакцинация	69
Макаров	Гигиеническая стрижка	55
Макаров	Профилактический осмотр	84
Макаров	Удаление новообразований в ротовой полости	55
Макаров	Лечение органов слуха	61
Макаров	Цитологическое исследование	62
Макаров	Диагностика органов слуха	62
Макаров	Консультация по изменению поведения животного	68
Макаров	Операция век	51
Макаров	Консультация по вопросам рациона	59
Макаров	Проведение ЭКГ	61
Макаров	Резекция поджелудочной железы	67
Макаров	Грыжесечение	56
Макаров	Наложение гипса	52
Макаров	Гормональные исследования	73
Макаров	Лечение от паразитов	64
Макаров	Перевязка	63
Макаров	Чипирование	58
Макаров	Клинический анализ крови	62
Макаров	Лечение мочевыделительной системы	70
Макаров	Установка капельницы	61
Макаров	Лечение переломов конечностей	49
Макаров	Проведение УЗИ	53
Макаров	Тест Ширмера	65
Макаров	Проведение МРТ	61
Макаров	Биохимия крови	65
Макаров	Эндоскопическая диагностика	62

Рис. 12: Результат выполнения запроса №6

EXPLAIN запроса:

id	select_type	table	paritions	type	possible_keys	key
1	SIMPLE	Doctors	NULL	ALL	PRIMARY	NULL
1	SIMPLE	Appointment_services	NULL	ref	Service_id,Doctor_id	Doctor_id
1	SIMPLE	Services	NULL	eq_ref	PRIMARY	PRIMARY

key_len	ref	rows	filtered	Extra
NULL	NULL	20	100.00	Using temporary
5	vetclinic.Doctors.Doctor_id	3281	100.00	Using where
4	vetclinic.Appointment_services.Service_id	1	100.00	NULL

Таблица 20: EXPLAIN для запроса №6

Объяснение запроса: с помощью группировки по врачам (Doctor_id) и услугам (Service_id) производится подсчет услуг для каждого врача.
На рисунке 13 приведена 3D-гистограмма, построенная по результатам запроса.

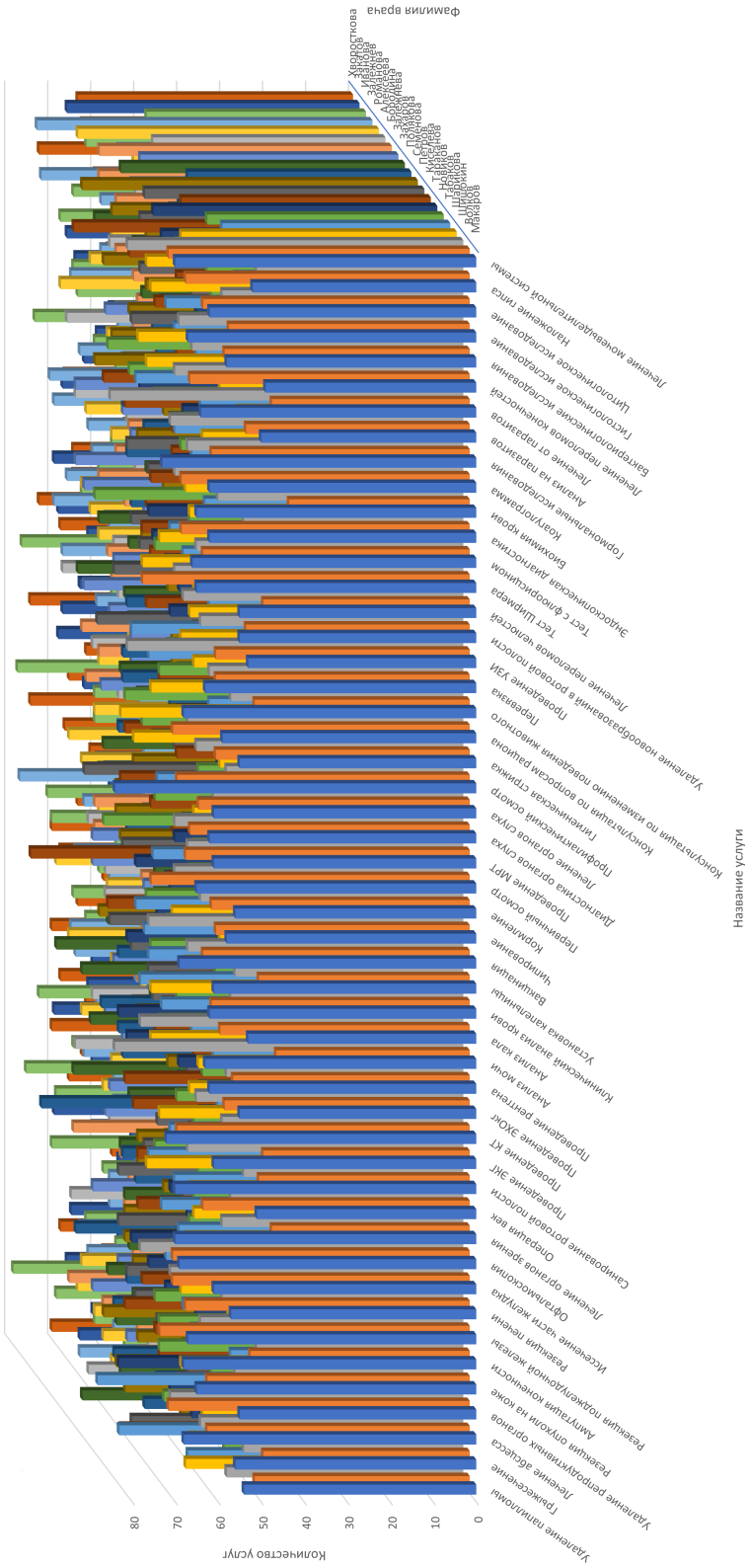


Рис. 13: Гистограмма к запросу №6

4.8 Запрос №7

Формулировка запроса: найти все товары, число продаж которых больше продаж товара A.
Текст запроса:

```
SELECT Product_name, count(*) AS Number_of_Sales,
(SELECT count(*) FROM Sales WHERE Product_id=3) AS Sales_of_product_A
FROM Sales JOIN Products
USING (Product_id)
GROUP BY Product_id
HAVING Number_of_sales > Sales_of_product_A;
```

На рисунке 14 приведен результат выполнения запроса. Выбрано 5 записей, время выполнения — 0.0083 с.

```
+-----------------------------------------+-----------------+--------------------+
| Product_name                            | Number_of_Sales | Sales_of_product_A |
+-----------------------------------------+-----------------+--------------------+
| Отоведин капли в уши                    |              68 |                 65 |
| Лосьон для глаз                         |              68 |                 65 |
| Анандин капли ушные                     |              67 |                 65 |
| Лосьон очищающий для ушей               |              66 |                 65 |
| Фунгивет для лечения грибковых заболеваний |              66 |                 65 |
+-----------------------------------------+-----------------+--------------------+
5 rows in set (0.0083 sec)
```

Рис. 14: Результат выполнения запроса №7

EXPLAIN запроса:

id	select_type	table	paritions	type	possible_keys	key
1	PRIMARY	Products	NULL	ALL	PRIMARY	NULL
1	PRIMARY	Sales	NULL	ref	Product_id	Product_id
2	SUBQUERY	Sales	NULL	ref	Product_id	Product_id

key_len	ref	rows	filtered	Extra
NULL	NULL	50	100.00	Using temporary
5	vetclinic.Products.Product_id	55	100.00	Using index
5	const	65	100.00	Using index

Таблица 21: EXPLAIN для запроса №7

Объяснение запроса: подзапрос находит количество продаж товара A (в данном случае это товар с id 3). Внешний запрос считает количество продаж каждого товара и оставляет записи, в которых количество продаж больше значения, найденного подзапросом (Sales_of_product_A).

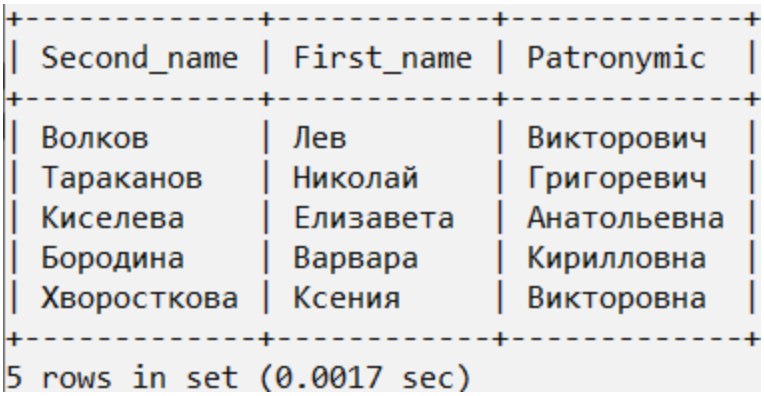
4.9 Запрос №8

Формулировка запроса: найти врачей, которые не оказывали услуги в стационаре пациенту A.

Текст запроса:

```
SELECT Second_name, First_name, Patronymic
FROM Doctors
WHERE Doctor_id NOT IN
(SELECT Doctor_id
FROM (Hospital_services JOIN Hospital
USING (Hospital_id))
WHERE Patient_id = 500);
```

На рисунке 15 приведен результат выполнения запроса. Выбрано 5 записей, время выполнения — 0.0017 с.



```
+-------------+------------+-------------+
| Second_name | First_name | Patronymic  |
+-------------+------------+-------------+
| Волков      | Лев        | Викторович  |
| Тараканов   | Николай    | Григоревич  |
| Киселева    | Елизавета  | Анатольевна |
| Бородина    | Варвара    | Кирилловна  |
| Хворосткова | Ксения     | Викторовна  |
+-------------+------------+-------------+
5 rows in set (0.0017 sec)
```

Рис. 15: Результат выполнения запроса №8

EXPLAIN запроса:

id	select_type	table	paritions	type	possible_keys	key
1	PRIMARY	Doctors	NULL	ALL	NULL	NULL
2	SUBQUERY	Hospital	NULL	ref	PRIMARY,Patient_id	Patient_id
2	SUBQUERY	Hospital_services	NULL	ref	Doctor_id,Hospital_id	Hospital_id

key_len	ref	rows	filtered	Extra
NULL	NULL	20	100.00	Using where
5	const	3	100.00	Using index
5	vetclinic.Hospital.Hospital_id	10	100.00	NULL

Таблица 22: EXPLAIN для запроса №8

Объяснение запроса: подзапрос находит врачей (Doctor_id), которые оказывали услуги пациенту A (в данном случае пациент с id 500). Внешний запрос выбирает из таблицы врачей (Doctors) записи, которые не соответствуют Doctor_id, найденным подзапросом.

Заключение

В результате выполнения курсовой работы были получены базовые навыки работы с SQL в системе управления базами данных MySQL.
В ходе работы была построена ER-диаграмма ветеринарной клиники, схема объектов из 9 сущностей: пациент, клиент, товар, ветеринарная карта, стационар, прием, услуга, врач, состояние.
По схеме объектов и ER-диаграмме была спроектирована база данных, которая позволяет хранить данные о продажах товаров, оказанных услугах на приеме и в стационаре, состояние пациентов в стационаре. База данных содержит 12 таблиц.
База данных ветеринарной клиники была заполнена с помощью программы на языке Java, код которой представлен в приложении Б. Всего база данных содержит 202 890 записей.
Было реализовано 8 запросов на языке SQL для спроектированной базы данных. Для двух запросов были построены гистограммы и для одного — 3D-гистограмма.

Приложение А. Скрипт генерации базы данных

```
DROP DATABASE IF EXISTS VetClinic;
CREATE DATABASE VetClinic;
CREATE TABLE VetClinic.Clients (
  Client_id INT PRIMARY KEY AUTO_INCREMENT,
  Second_name VARCHAR(30),
  First_name VARCHAR(30),
  Patronymic VARCHAR(35),
  Phone_number VARCHAR(20));
CREATE TABLE VetClinic.Products (
  Product_id INT PRIMARY KEY AUTO_INCREMENT,
  Product_name VARCHAR(50),
  Price DECIMAL(6,2));
CREATE TABLE VetClinic.Sales (
  Sale_id INT PRIMARY KEY AUTO_INCREMENT,
  Client_id INT,
  Product_id INT,
  Sale_date DATE,
  FOREIGN KEY(Client_id) REFERENCES VetClinic.Clients(Client_id)
  ON DELETE restrict ON UPDATE restrict,
  FOREIGN KEY(Product_id) REFERENCES VetClinic.Products(Product_id)
  ON DELETE restrict ON UPDATE restrict);
CREATE TABLE VetClinic.Patients (
  Patient_id INT PRIMARY KEY AUTO_INCREMENT,
  Owner_id INT,
  Moniker VARCHAR(30),
  Birthday DATE,
  Animal_kind VARCHAR(20),
  Gender CHAR(1),
  FOREIGN KEY(Owner_id) REFERENCES VetClinic.Clients(Client_id)
  ON DELETE restrict ON UPDATE restrict);
CREATE TABLE VetClinic.Doctors (
  Doctor_id INT PRIMARY KEY AUTO_INCREMENT,
  Second_name VARCHAR(30),
  First_name VARCHAR(30),
  Patronymic VARCHAR(35),
  Phone_number VARCHAR(20));
CREATE TABLE VetClinic.Appointments (
  Appointment_id INT PRIMARY KEY AUTO_INCREMENT,
  Patient_id INT,
  Appointment_date DATE,
  Appointment_time TIME,
  FOREIGN KEY(Patient_id) REFERENCES VetClinic.Patients(Patient_id)
  ON DELETE restrict ON UPDATE restrict);
CREATE TABLE VetClinic.Services (
  Service_id INT PRIMARY KEY AUTO_INCREMENT,
  Service_name VARCHAR(50),
```

```
  Price DECIMAL(6,2));
CREATE TABLE VetClinic.Appointment_services (
  id INT PRIMARY KEY AUTO_INCREMENT,
  Doctor_id INT,
  Appointment_id INT,
  Service_id INT,
  FOREIGN KEY(Doctor_id) REFERENCES VetClinic.Doctors(Doctor_id)
  ON DELETE restrict ON UPDATE restrict,
 FOREIGN KEY(Appointment_id) REFERENCES VetClinic.Appointments(Appointment_id)
  ON DELETE restrict ON UPDATE restrict,
  FOREIGN KEY(Service_id) REFERENCES VetClinic.Services(Service_id)
  ON DELETE restrict ON UPDATE restrict);
CREATE TABLE VetClinic.Hospital (
  Hospital_id INT PRIMARY KEY AUTO_INCREMENT,
  Patient_id INT,
  Admission_date DATE,
  Discharge_date DATE,
  FOREIGN KEY(Patient_id) REFERENCES VetClinic.Patients(Patient_id)
  ON DELETE restrict ON UPDATE restrict);
CREATE TABLE VetClinic.Hospital_services (
  id INT PRIMARY KEY AUTO_INCREMENT,
  Doctor_id INT,
  Hospital_id INT,
  Service_id INT,
  FOREIGN KEY(Doctor_id) REFERENCES VetClinic.Doctors(Doctor_id)
  ON DELETE restrict ON UPDATE restrict,
  FOREIGN KEY(Hospital_id) REFERENCES VetClinic.Hospital(Hospital_id)
  ON DELETE restrict ON UPDATE restrict,
  FOREIGN KEY(Service_id) REFERENCES VetClinic.Services(Service_id)
  ON DELETE restrict ON UPDATE restrict);
CREATE TABLE VetClinic.Features (
  Feature_id INT PRIMARY KEY AUTO_INCREMENT,
  Feature_name VARCHAR(50));
CREATE TABLE VetClinic.Conditions (
  Condition_id INT PRIMARY KEY AUTO_INCREMENT,
  Doctor_id INT,
  Hospital_id INT,
  Feature_id INT,
  Feature_value VARCHAR(50),
  Fixation_date DATE,
  Fixation_time TIME,
  FOREIGN KEY(Doctor_id) REFERENCES VetClinic.Doctors(Doctor_id)
  ON DELETE restrict ON UPDATE restrict,
  FOREIGN KEY(Hospital_id) REFERENCES VetClinic.Hospital(Hospital_id)
  ON DELETE restrict ON UPDATE restrict,
  FOREIGN KEY(Feature_id) REFERENCES VetClinic.Features(Feature_id)
  ON DELETE restrict ON UPDATE restrict);
```

Приложение Б. Программа заполнения таблицы

Файл Generator.java

```
import java.util.Random;

public class Generator {
  Random random = new Random();
  final static String[] maleNames = {"Александр", "Алексей", "Анатолий", "Андрей", "Артем",
    "Борис", "Валерий", "Василий", "Виктор", "Виталий", "Владимир", "Георгий", "Глеб",
   "Григорий", "Дмитрий", "Евгений", "Егор", "Иван", "Илья", "Кирилл", "Лев", "Максим",
   "Михаил", "Николай", "Никита", "Олег", "Петр", "Семен", "Сергей", "Юрий"};
  final static String[] femaleNames = {"Александра", "Алиса", "Анастасия", "Анна", "Варвара",
   "Валерия", "Вера", "Вероника", "Дарья", "Евгения", "Екатерина", "Елена", "Елизавета",
   "Ирина", "Кристина", "Ксения", "Любовь", "Людмила", "Маргарита", "Марина", "Мария",
   "Надежда", "Нина", "Оксана", "Ольга", "Полина", "Светлана", "София", "Татьяна"};
  final static String[] surnames = {"Иванов", "Смирнов", "Кузнецов", "Васильев", "Петров",
   "Соколов", "Михайлов", "Новиков", "Федоров", "Морозов", "Волков", "Алексеев", "Лебедев",
   "Семенов", "Хаборов", "Хворостков", "Егоров", "Павлов", "Степанов", "Николаев", "Орлов",
   "Макаров", "Никитин", "Работин", "Решетов", "Цаплин", "Зайцев", "Захаров", "Соловьев",
   "Борисов", "Яковлев", "Романов", "Воробьев", "Петелин", "Петухов", "Фролов", "Королев",
   "Гусев", "Киселев", "Поляков", "Сорокин", "Виноградов", "Демидов", "Шариков", "Бородин",
   "Ошурков", "Червяков", "Сидоров", "Табаков", "Тараканов", "Назимов", "Ларин", "Лапин",
   "Гончаров", "Шишокин", "Шишкин", "Закатов", "Залежнев", "Варенников", "Валов"};
  final static String[] maleMonikers = {"Айс", "Альф", "Арто", "Арчи", "Барни", "Барон",
   "Вилли", "Билли", "Дилли", "Бруно", "Бинго", "Локи", "Феликс", "Джек", "Фред", "Микки",
   "Чейз", "Кейк", "Лука", "Буян", "Граф", "Ник", "Честер", "Хэнк", "Тедди", "Тинг", "Рик",
   "Дюк", "Люк", "Пип", "Рекс", "Туз", "Лео", "Уголек"};
  final static String[] femaleMonikers = {"Рокси", "Бони", "Ава", "Берта", "Ленни", "Милка",
   "Маркиза", "Дора", "Ромашка", "Полли", "Пенелопа", "Салли", "Оливия", "Трикси",
   "Стелла", "Лея", "Руби", "Фокси", "Дымка", "Фиона", "Симона", "Тесса", "Лили", "Лулу",
   "Кэт", "Дина", "Кора", "Фея", "Снежка", "Ночка", "Клякса"};
  final static String[] animalType = {"кошка", "собака", "попугай", "хомяк", "кролик", "шиншила"};
  final static String[] patronymic = {"Александров", "Алексеев", "Анатольев", "Андреев",
   "Артемов", "Борисов", "Валерьев", "Васильев", "Викторов", "Витальев", "Владиров",
   "Георгиев", "Глебов", "Григорев", "Дмитриев", "Евгеньев", "Егоров", "Иванов", "Кириллов",
   "Львов", "Михайлов", "Николаев", "Олегов", "Петров", "Семенов", "Сергеев", "Юрьев"};
  final static String[] goods = {"Отоведин капли в уши", "КотЭрвин раствор для приема внутрь",
   "Ветмедин S таблетки", "Марфлоксин таблетки", "Ошейник от блох, клещей для собак",
   "Милпразон антигельминтик", "Мильбемакс антигельминтик",
   "Вакдерм вакцина суспензия для инъекций", "Микродерм вакцина для животных",
   "АктиВет таблетки", "Минеральный комплекс", "Сера для животных",
   "Комплекс для кожи и шерсти", "Код Омега Плюс", "Лактобифадол пробиотик",
   "Травка для кошек лоток", "Травка для кошек пакет", "Катобевит", "Эвинтон раствор",
   "Лосьон для глаз", "Максидин капли глазные и интерназальные",
   "Средство для удаления слезных пятен", "Ирис капли глазные", "Лосьон очищающий для глаз",
   "Анандин капли ушные", "Лосьон очищающий для ушей", "Лосьон для ушей с фитокомплексом",
   "Отибиовин капли ушные", "Отодепин капли в уши с маслом сосны",
   "Чистые ушки лосьон для очистки ушей", "Рикарфа таблетки", "Хондартон раствор для инъекций",
   "Ветом 3 порошок", "Диаркан брикеты-сахарные кубики", "Молочная кислота 40%",
   "Веракол раствор для инъекций", "Ветсорбин таблетки", "Гепатолюкс для печени",
   "Средство для дезинфекции", "Онсиор таблетки", "Мелоксидил суспензия",
   "Ципровет таблетки", "Септогель", "Фунгивет для лечения грибковых заболеваний",
   "Фунгин Форте раствор для наружного применения", "Корм", "Сульф 120 таблетки",
   "Амоксоил ретард суспензия для инъекций", "Синулокс шприц", "Арбимектин таблетки"};
```

```
  final static String[] services = {"Удаление папилломы", "Грыжесечение", "Лечение абсцесса",
   "Удаление репродуктивных органов", "Резекция опухоли на коже", "Ампутация конечности",
   "Резекция поджелудочной железы", "Резекция печени", "Иссечение части желудка",
   "Офтальмоскопия", "Лечение органов зрения", "Операция век",
   "Санирование ротовой полости", "Проведение ЭКГ", "Проведение КТ",
   "Проведение ЭХОкг", "Проведение рентгена", "Анализ мочи", "Анализ кала",
   "Клинический анализ крови ", "Установка капельницы", "Вакцинация", "Чипирование",
   "Кормление", "Первичный осмотр", "Проведение МРТ", "Диагностика органов слуха",
   "Лечение органов слуха", "Профилактический осмотр", "Гигиеническая стрижка",
   "Консультация по вопросам рациона", "Консультация по изменению поведения животного", "Перевязка",
   "Проведение УЗИ", "Удаление новообразований в ротовой полости", "Лечение переломов челюстей",
   "Тест Ширмера", "Тест с флюорисцином", "Эндоскопическая диагностика", "Биохимия крови",
   "Коагулограмма", "Гормональные исследования", "Анализ на паразитов",
   "Лечение от паразитов", "Лечение переломов конечностей", "Бактериологические исследования",
   "Гистологическое исследование", "Цитологическое исследование", "Наложение гипса",
   "Лечение мочевыделительной системы"};
  final static String[] featuresDigit = {"Температура", "Артериальное давление", "Вес",
   "Частота пульса", "Частота дыхания"};
  final static String[] featuresNonDigit = {"Состояние ротовой полости", "Состояние шерсти/перьев",
   "Состояние кожи", "Состояние глаз", "Состояние верхних конечностей",
   "Состояние нижних конечностей", "Состояние ушей", "Состояние хвоста", "Аппетит",
   "Состояние когтей", "Состояние брюшной полости", "Состояние подкожных лимфоузлов",
   "Состояние слизистых оболочек", "Состояние грудной клетки", "Общее состояние"};
final static String[] featureValue = {"Критическое", "Удовлетворительное", "Ниже нормы", "Норма"};

  private String addDays(String date, int days) { //добавление дней к дате
   var ymd = date.split("-");
   int day = Integer.parseInt(ymd[2]) + days;
   int month = Integer.parseInt(ymd[1]);
   int year = Integer.parseInt(ymd[0]);
   if (month == 4 || month == 6 || month == 9 || month == 11) {
    if (day >= 30) {
     day -= 30;
     month++;
    }
   } else if (month == 2) {
    if (year % 4 == 0) {
     if (day >= 29) {
      day -= 29;
      month++;
     }
    } else {
     if (day >= 28) {
      day -= 28;
      month++;
     }
    }
   } else {
    if (day >= 31) {
     day -= 31;
     month++;
     if (month == 13) {
     month = 1;
     year++;
```

```
   }
  }
 }
 return year + "-" + month + "-" + day;
}

public String randomPhone() {
 StringBuilder phone = new StringBuilder();
 phone.append("+7(");
 for (int i = 0; i < 3; i++)
  phone.append(random.nextInt(9));
 phone.append(")");
 for (int i = 0; i < 7; i++) {
  phone.append(random.nextInt(9));
  if (i == 2 || i == 4)
   phone.append("-");
 }
 return phone.toString();
}

public String randomDayMonth(int year) {
 StringBuilder date = new StringBuilder();
 int month = random.nextInt(1, 12);
 date.append(month);
 date.append("-");
 int day;
 if (month == 2) {
  if (year % 4 == 0)
   day = random.nextInt(1, 29);
  else day = random.nextInt(1, 28);
 } else if (month == 4 || month == 6 || month == 9 || month == 11) {
  day = random.nextInt(1, 30);
 } else {
  day = random.nextInt(1, 31);
 }
 date.append(day);
 return date.toString();
}

public String randomBirthday() {
 StringBuilder birthday = new StringBuilder();
 int year = random.nextInt(2000, 2021);
 birthday.append(year);
 birthday.append("-");
 birthday.append(randomDayMonth(year));
 return birthday.toString();
}

public String[] randomHuman() {
 String[] result = new String[4];
 if (random.nextBoolean()) {
  result[0] = surnames[random.nextInt(surnames.length - 1)];
  result[1] = maleNames[random.nextInt(maleNames.length - 1)];
  result[2] = patronymic[random.nextInt(patronymic.length - 1)] + "ич";
```

```
 } else {
  result[0] = surnames[random.nextInt(surnames.length - 1)] + "а";
  result[1] = femaleNames[random.nextInt(femaleNames.length - 1)];
  result[2] = patronymic[random.nextInt(patronymic.length - 1)] + "на";
 }
 result[3] = randomPhone();
 return result;
}

public String[] randomAnimal() {
 String[] result = new String[4];
 if (random.nextBoolean()) {
  result[0] = maleMonikers[random.nextInt(maleMonikers.length - 1)];
  result[1] = "М";
 } else {
  result[0] = femaleMonikers[random.nextInt(femaleMonikers.length - 1)];
  result[1] = "Ж";
 }
 result[2] = randomBirthday();
 result[3] = animalType[random.nextInt(animalType.length - 1)];
 return result;
}

public String[] randomPrice(int i) {
 String[] result = new String[2];
 result[0] = goods[i];
 result[1] = String.format("%d.%d", random.nextInt(200, 9999), random.nextInt(99));
 return result;
}

public String[] randomAppointment() {
 String[] result = new String[4];
 int year = random.nextInt(2019, 2021);
 result[0] = year + "-" + randomDayMonth(year);
 result[1] = random.nextInt(8, 20) + ":00";
 return result;
}

public String[] randomSale() {
 String[] result = new String[2];
 int year = random.nextInt(2019, 2021);
 result[0] = year + "-" + randomDayMonth(year);
 result[1] = String.valueOf(random.nextInt(1, goods.length + 1));
 return result;
}

public String[] randomHospital() {
 String[] result = new String[3];
 int year = random.nextInt(2019, 2021);
 result[0] = year + "-" + randomDayMonth(year);
 int days = random.nextInt(5, 30);
 result[1] = addDays(result[0], days);
 result[2] = String.valueOf(days);
 return result;
```

```
  }

  public String[] randomCondition(int doctorNumber, int days, String start, boolean type)
  {//type - true легкое животное
   String[] result = new String[5];
   int idx = random.nextInt(19);
   result[0] = String.valueOf(idx + 1);
   if (idx < featuresDigit.length) {
    if (idx == 0) {
     result[1] = String.format("%d.%d", random.nextInt(34, 45), random.nextInt(9));
    } else if (idx == 1) {
     result[1] = String.format("%d-%d", random.nextInt(100, 200), random.nextInt(50, 150));
    } else if (idx == 2) {
     if (type) {
      result[1] = random.nextInt(200, 2000) + " г";
     } else {
      result[1] = random.nextInt(1, 20) + " кг";
     }
    } else {
     result[1] = random.nextInt(20, 150) + " в мин";
    }
   } else {
    result[1] = featureValue[random.nextInt(featureValue.length - 1)];
   }
   result[2] = addDays(start, random.nextInt(days));
   result[3] = String.format("%d:%d", random.nextInt(23), random.nextInt(59));
   result[4] = String.valueOf(random.nextInt(1, doctorNumber + 1));
   return result;
  }

  public String[] randomService(int i) {
   String[] result = new String[2];
   result[0] = services[i];
   result[1] = String.format("%d.%d", random.nextInt(500, 10000), random.nextInt(100));
   return result;
  }

  public String[] randomServices(int doctors) {
   String[] result = new String[2];
   result[0] = String.valueOf(random.nextInt(1, doctors + 1));
   result[1] = String.valueOf(random.nextInt(1, services.length + 1));
   return result;
  }
}
```

Файл Filler.java

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class Filler {
  final static int clientNumber = 500; //количество клиентов
  final static int doctorNumber = 20; //количество врачей
  final static int minAnimals = 1; //минимальное количество животных у одного клиента
```

```
  final static int maxAnimals = 5; //максимальное количество животных у одного клиента
  final static int minGoods = 1; //минимальное количество покупок у одного клиента
  final static int maxGoods = 10; //максимальное количество покупок у одного клиента
  final static int patientNumber = clientNumber*(minAnimals+maxAnimals)/2;//количество пациентов
  final static int minAppointments = 5; //минимальное количество приемов одного пациента
  final static int maxAppointments = 10; //максимальное количество приемов одного пациента
  final static int appointNumber = patientNumber*(minAppointments+maxAppointments)/2;
  //количество приемов
  final static int minHospital = 1; //минимальное количество попаданий в стационар одного пациента
  final static int maxHospital = 8; //максимальное количество попаданий в стационар одного пациента
  final static int minCondition = 5; //минимальное количество состояний в стационаре
  final static int maxCondition = 10; //максимальное количество состояний в стационаре
  final static int minServicesAppointment = 1; //минимальное количество услуг на приеме
  final static int maxServicesAppointment = 10; //максимальное количество услуг на приеме
  final static int minServicesHospital = 5; //минимальное количество услуг в стационаре
  final static int maxServicesHospital = 15; //максимальное количество услуг в стационаре
  static List<Integer> daysInHospital = new ArrayList<>();
  static List<String> patientType = new ArrayList<>();
  static List<String> startInHospital = new ArrayList<>();
  static List<Integer> patientInHospital = new ArrayList<>();
  public static void main(String[] args) {
   Generator generator = new Generator();
   String url = "jdbc:mysql://127.0.0.1/vetclinic";
   String userName = "root";
   String pass = "Windows_10Microsoft";
   try (Connection connection = DriverManager.getConnection(url, userName, pass)) {
    Statement statement = connection.createStatement();
//клиенты
    StringBuilder request = new StringBuilder(
     "INSERT INTO clients (Second_name,First_name,Patronymic,Phone_number) VALUES ");
    for(int i = 0; i < clientNumber; i++) {
     var client = generator.randomHuman();
     request.append(String.format("('%s','%s','%s','%s'),",
     client[0], client[1], client[2], client[3]));
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица клиентов заполнена");
//товары
    request = new StringBuilder("INSERT INTO products (Product_name,Price) VALUES ");
    for(int i = 0; i < Generator.goods.length; i++) {
     var product = generator.randomPrice(i);
     request.append(String.format("('%s',%s),",
     product[0], product[1]));
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица товаров заполнена");
//врачи
    request = new StringBuilder(
     "INSERT INTO doctors (Second_name,First_name,Patronymic,Phone_number) VALUES ");
    for(int i = 0; i < doctorNumber; i++) {
     var doctor = generator.randomHuman();
     request.append(String.format(
```

```
     "('%s','%s','%s','%s'),",
     doctor[0], doctor[1], doctor[2], doctor[3]));
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица врачей заполнена");
//признаки
    request = new StringBuilder("INSERT INTO features (Feature_name) VALUES ");
    for(int i = 0; i < Generator.featuresDigit.length;i++) {
     request.append(String.format("('%s'),",
     Generator.featuresDigit[i]));
    }
    for(int i = 0; i < Generator.featuresNonDigit.length;i++) {
     request.append(String.format("('%s'),",
     Generator.featuresNonDigit[i]));
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица признаков заполнена");
//услуги
    request = new StringBuilder("INSERT INTO services (Service_name,Price) VALUES ");
    for(int i = 0; i < Generator.services.length; i++) {
     var service = generator.randomService(i);
     request.append(String.format("('%s',%s),",service[0],service[1]));
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица услуг заполнена");
//пациенты
    request = new StringBuilder(
      "INSERT INTO patients (Moniker,Gender,Birthday,Animal_kind,Owner_id) VALUES ");
    int animalNumber = minAnimals;
    int range = clientNumber/(maxAnimals - minAnimals + 1);
    for(int i = 1; i <= clientNumber; i++) {
     for (int j = 0; j < animalNumber; j++) {
      var patient = generator.randomAnimal();
      request.append(String.format("('%s','%s','%s','%s',%s),",
      patient[0], patient[1], patient[2], patient[3],i));
      patientType.add(patient[3]);
     }
     if(i % range == 0)
     animalNumber++;
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица пациентов заполнена");
//продажи
    request = new StringBuilder(
      "INSERT INTO sales (Sale_date,Product_id,Client_id) VALUES");
    int goodNumber = minGoods;
    range = clientNumber/(maxGoods - minGoods + 1);
    for(int i = 1; i <= clientNumber; i++) {
     for(int j = 0; j < goodNumber; j++) {
      var sale = generator.randomSale();
```

```
      request.append(String.format("('%s',%s,%s),", sale[0], sale[1], i));
     }
     if(i % range == 0)
      goodNumber++;
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица продаж заполнена");
//приемы
    request = new StringBuilder(
     "INSERT INTO appointments (Appointment_date,Appointment_time,Patient_id) VALUES ");
    int appointmentNumber = minAppointments;
    range = patientNumber/(maxAppointments - minAppointments + 1);
    for(int i = 1; i <= patientNumber; i++) {
     for (int j = 0; j < appointmentNumber; j++) {
      var appointment = generator.randomAppointment();
      request.append(String.format("('%s','%s',%s),",
      appointment[0], appointment[1],i));
     }
     if(i % range == 0)
      appointmentNumber++;
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица приемов заполнена");
//стационар
    request = new StringBuilder(
     "INSERT INTO hospital (Admission_date,Discharge_date,Patient_id) VALUES ");
    int hospitalNumber = minHospital;
    range = patientNumber/(maxHospital-minHospital+1);
    int current = 0; //количество уже добавленных записей
    for(int i = 1; i < patientNumber; i++) {
     for (int j = 0; j < hospitalNumber; j++) {
      var hospital = generator.randomHospital();
      request.append(String.format("('%s','%s',%s),",
      hospital[0], hospital[1],i));
      daysInHospital.add(Integer.parseInt(hospital[2]));
      startInHospital.add(hospital[0]);
      patientInHospital.add(i-1);
     }
     if((hospitalNumber % 2) == 1 || hospitalNumber == 2) {
     if(i - current == range) {
      hospitalNumber++;
      current += range;
     }
     } else if(i - current == range + 1) {
       hospitalNumber++;
       current += range + 1;
      }
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица стационара заполнена");
//состояния
```

```
    request = new StringBuilder("INSERT INTO conditions"+
     "(Feature_id,Feature_value,Fixation_date,Fixation_time,Doctor_id,Hospital_id) VALUES ");
    int conditionNumber = minCondition;
    range = daysInHospital.size()/(maxCondition-minCondition+1);
    for(int i = 0; i < daysInHospital.size(); i++) {
     String patient = patientType.get(patientInHospital.get(i));
     boolean type = !(patient.equals("собака") || patient.equals("кошка"));
     for(int j = 0; j < conditionNumber; j++) {
      var condition = generator.randomCondition(doctorNumber,daysInHospital.get(i),
      startInHospital.get(i),type);
      request.append(String.format("(%s,'%s','%s','%s',%s,%s),",
      condition[0],condition[1],condition[2],condition[3],condition[4],i+1));
     }
     if((i+1) % range == 0)
      conditionNumber++;
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица состояний заполнена");
//услуги на приеме
    request = new StringBuilder(
     "INSERT INTO Appointment_services (Doctor_id,Service_id,Appointment_id) VALUES ");
    int serviceNumber = minServicesAppointment;
    range = appointNumber/(maxServicesAppointment - minServicesAppointment + 1);
    for(int i = 1; i <= appointNumber; i++) {
     for(int j = 0; j < serviceNumber; j++) {
      var service = generator.randomServices(doctorNumber);
      request.append(String.format("(%s,%s,%s),", service[0],service[1],i));
     }
     if(i % range == 0)
     serviceNumber++;
    }
    request.deleteCharAt(request.length()-1);
    statement.executeUpdate(request.toString());
    System.out.println("Таблица услуг на приеме заполнена");
//услуги в стационаре
    request = new StringBuilder(
     "INSERT INTO Hospital_services (Doctor_id,Service_id,Hospital_id) VALUES ");
    serviceNumber = minServicesHospital;
    range = daysInHospital.size()/(maxServicesHospital - minServicesHospital + 1);
    current = 0;
    for(int i = 1; i <= daysInHospital.size(); i++) {
     for(int j = 0; j < serviceNumber; j++) {
      var service = generator.randomServices(doctorNumber);
      request.append(String.format("(%s,%s,%s),", service[0],service[1],i));
     }
     if(serviceNumber % 2 == 1 || serviceNumber == 10) {
      if(i - current == range + 1) {
       serviceNumber++;
       current += range + 1;
      }
     } else if(i - current == range) {
       serviceNumber++;
       current+=range;
```

```
      }
     }
     request.deleteCharAt(request.length()-1);
     statement.executeUpdate(request.toString());
     System.out.println("Таблица услуг в стационаре заполнена");
    } catch (SQLException e) {
     e.printStackTrace();
    }
   }
}
```