**Итоговая контрольная работа**

**Информация о проекте**

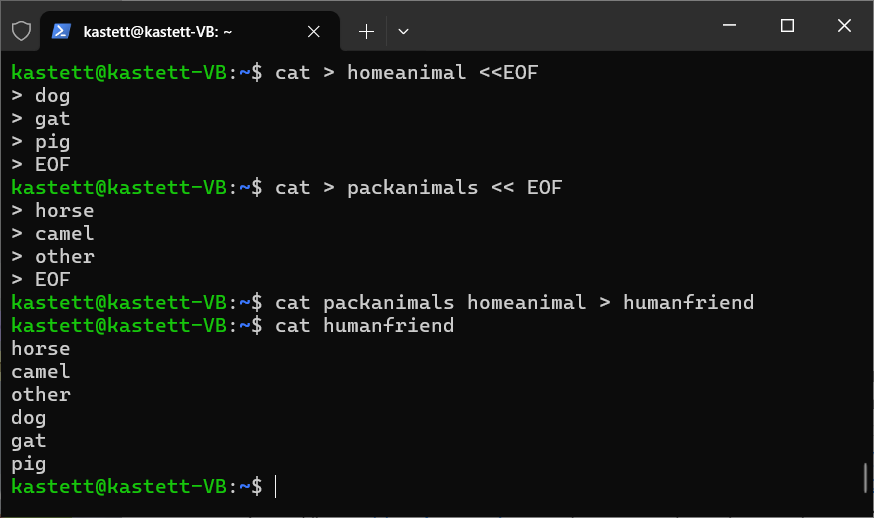
Необходимо организовать систему учета для питомника в котором живут домашние и вьючные животные.

**Как сдавать проект**

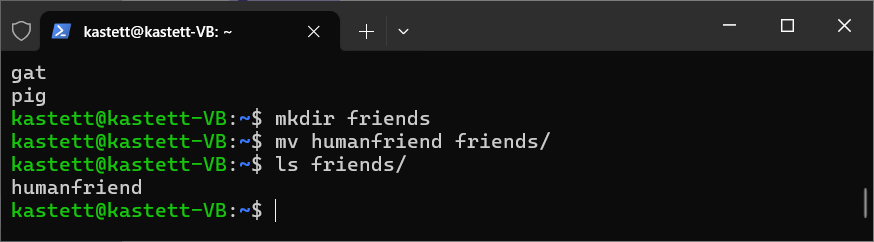
Для сдачи проекта необходимо создать отдельный общедоступный репозиторий (Github, gitlub, или Bitbucket). Разработку вести в этом репозитории, использовать пул реквесты на изменения. Программа должна запускаться и работать, ошибок при выполнении программы быть не должно. Программа, может использоваться в различных системах, поэтому необходимо разработать класс в виде конструктора

**Задание**

1. Используя команду **cat** в терминале операционной системы **Linux**, создать два файла Домашние животные **“homeanimal”** (заполнив файл собаками, кошками, хомяками) и Вьючные животными **“packanimals”** заполнив файл Лошадьми, верблюдами и ослы), а затем объединить их. Просмотреть содержимое созданного файла. Переименовать файл, дав ему новое имя Друзья человека **“humanfriend”.**

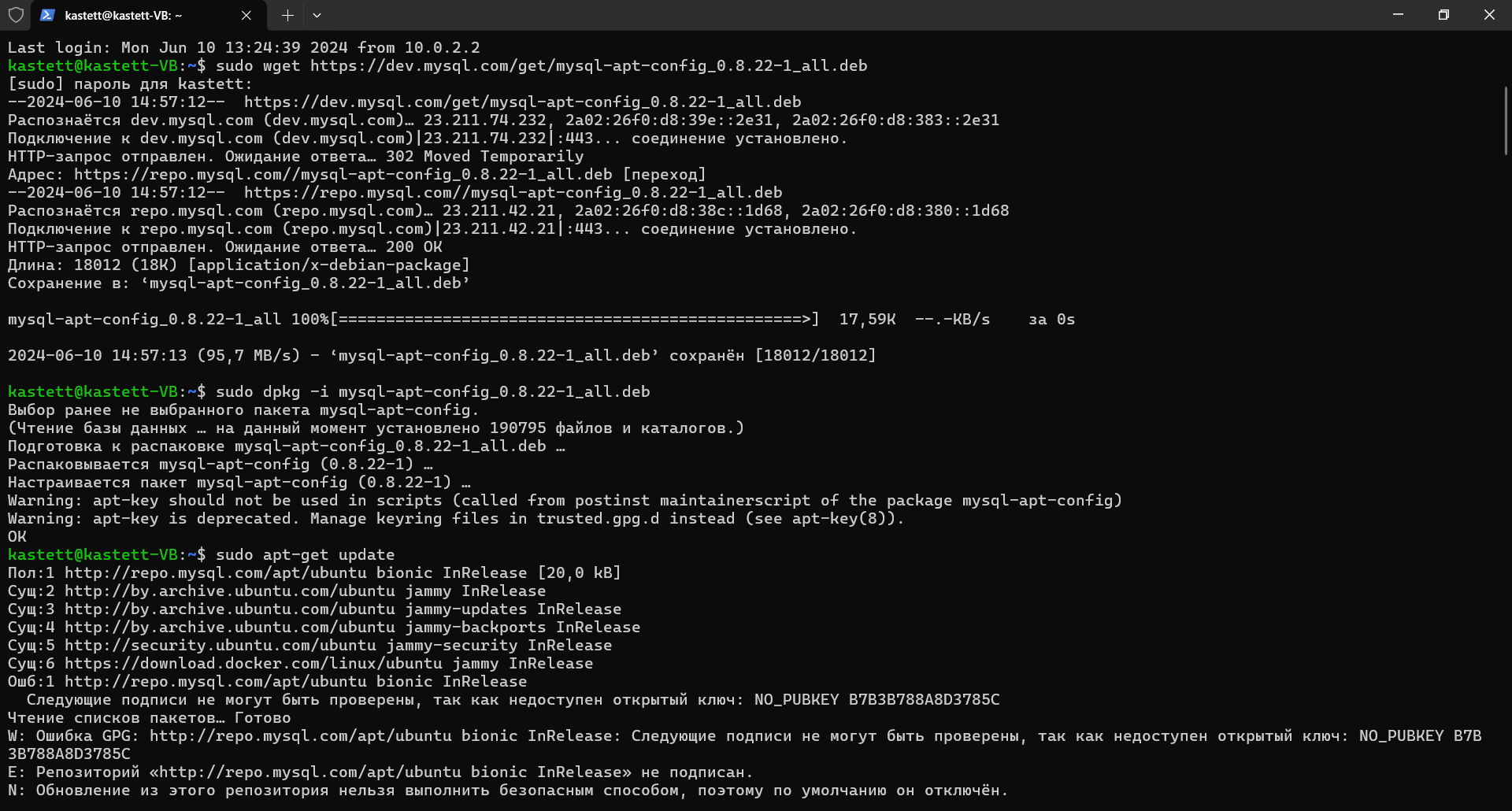


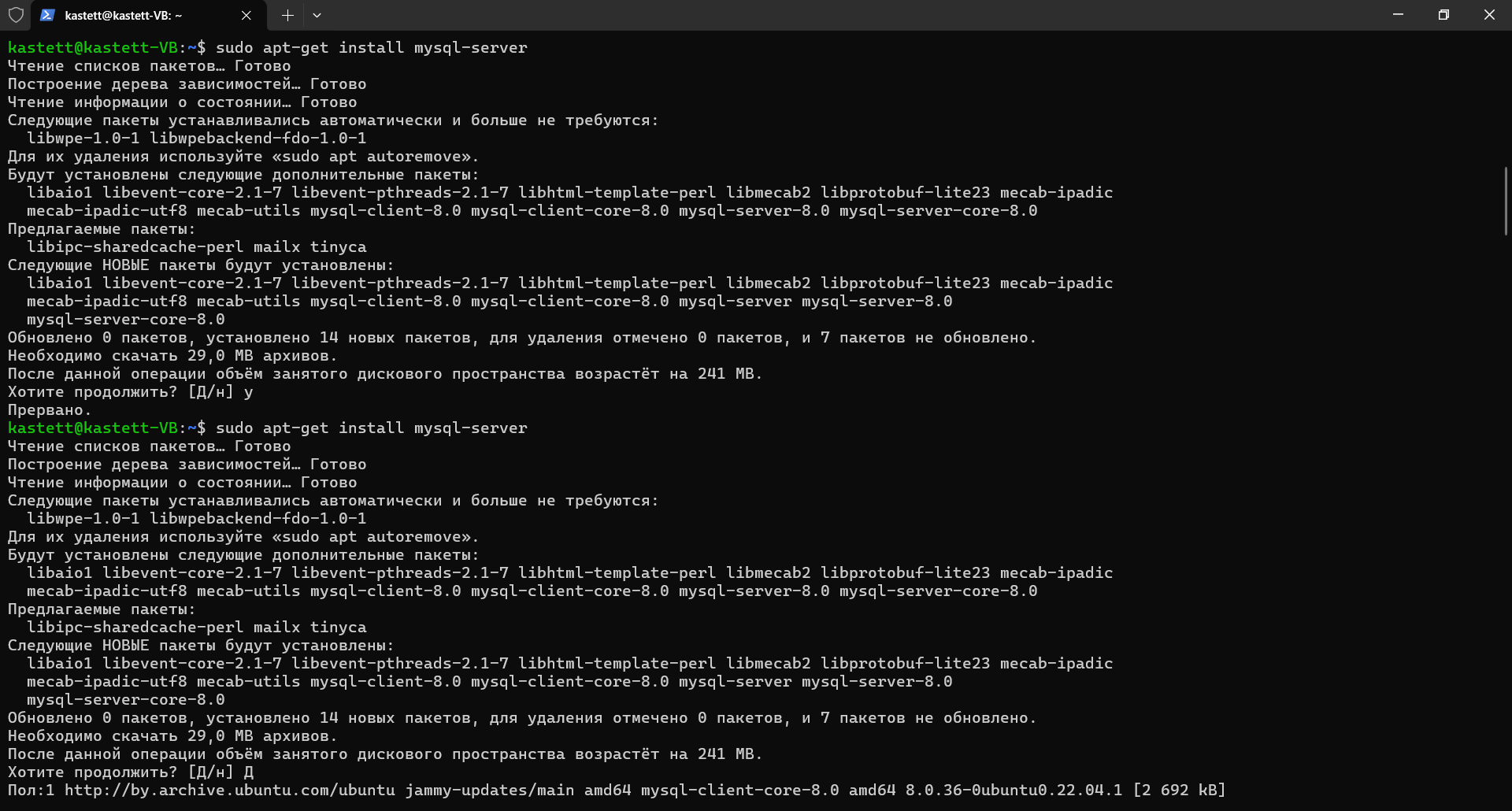
2. Создать директорию, переместить файл туда.

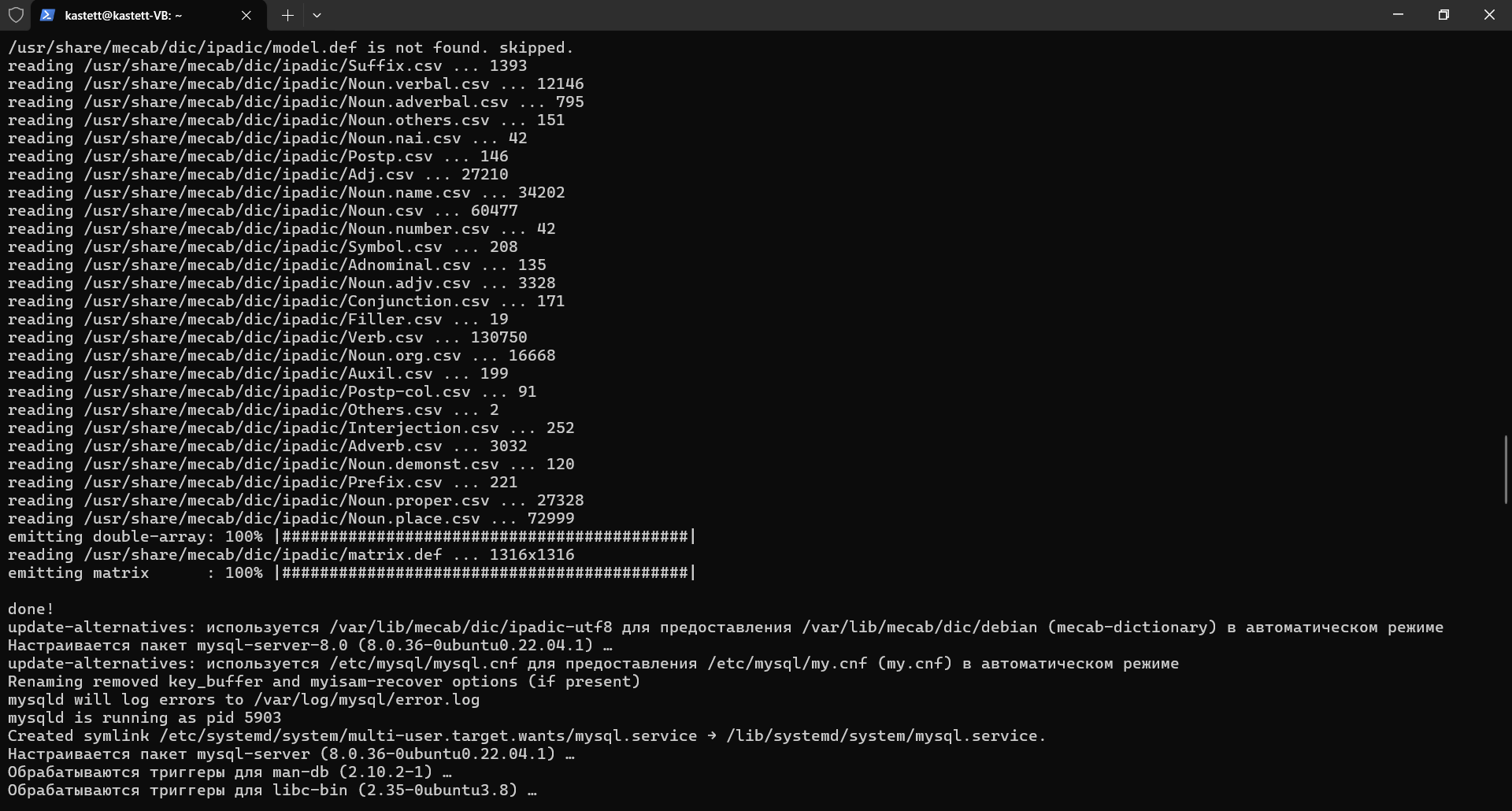


3. Подключить дополнительный репозиторий MySQL. Установить любой пакет

из этого репозитория.



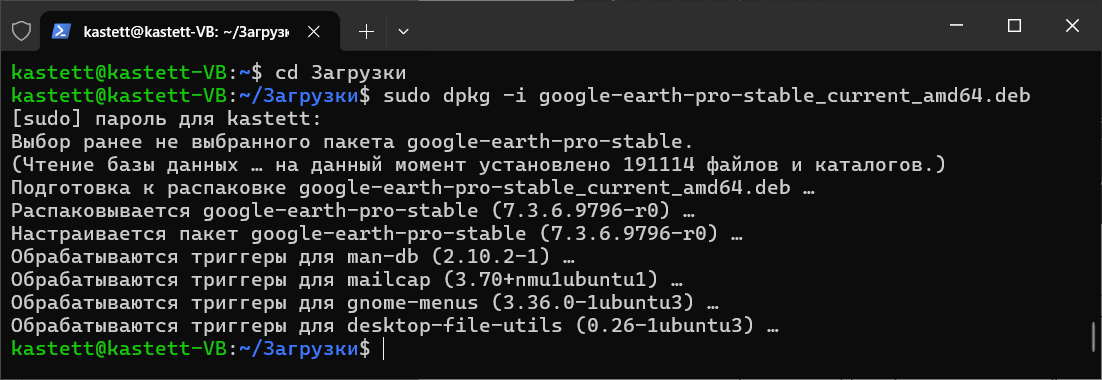


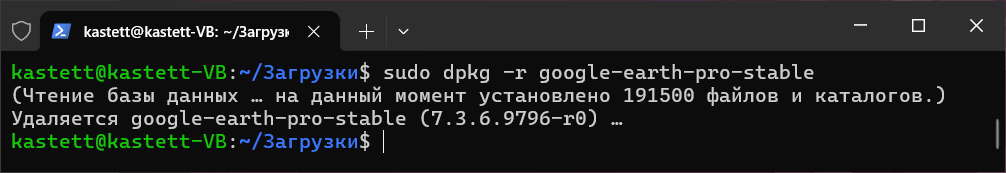


4. Установить и удалить deb-пакет с помощью dpkg.

Скачал deb пакет google-earth-pro-stable\_current\_amd64.deb с официального сайта

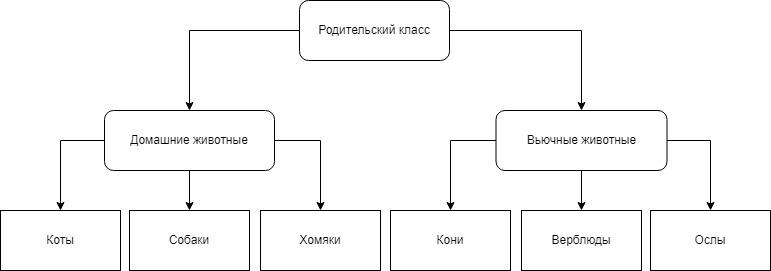
https://dl.google.com/dl/earth/client/current/google-earth-pro-stable\_current\_amd64.deb





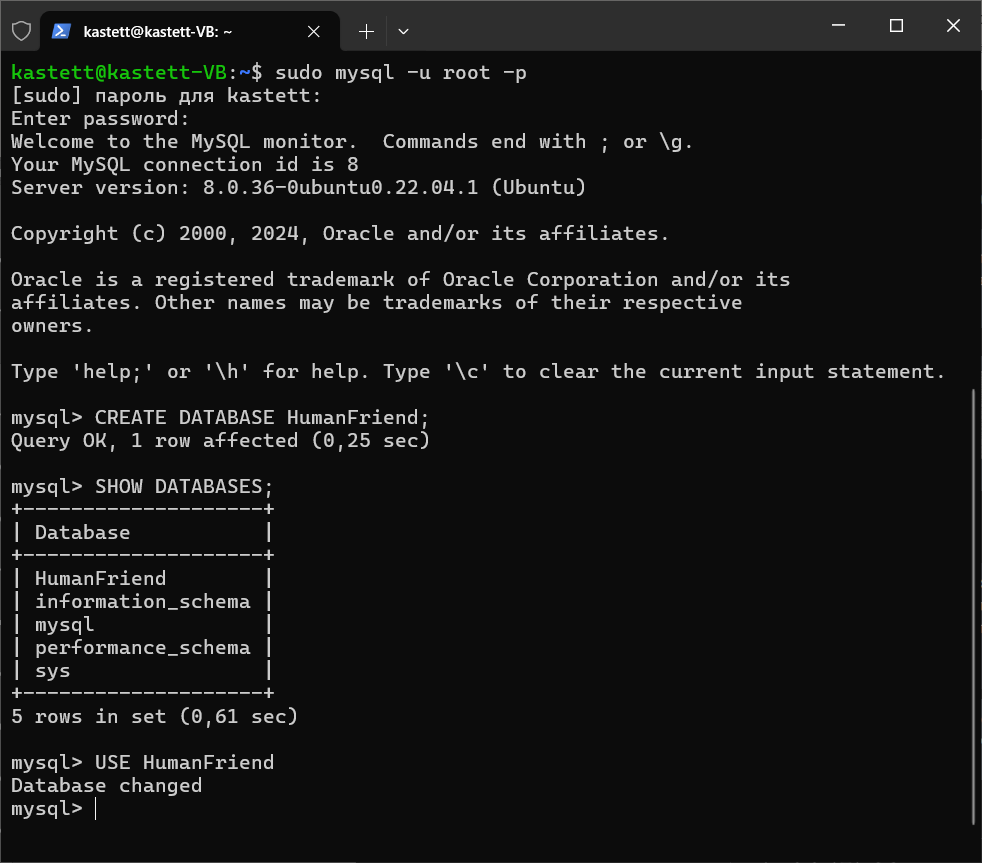
5. Выложить историю команд в терминале ubuntu (Сделано)

6. Нарисовать диаграмму, в которой есть класс «родительский класс», «домашние животные» и «вьючные животные», в составы которых в случае домашних животных войдут классы: собаки, кошки, хомяки, а в класс вьючные животные войдут: лошади, верблюды и ослы.



7. В подключенном MySQL репозитории создать базу данных “Друзья

человека” - **HumanFriend**



8. Создать таблицы с иерархией из диаграммы в БД

**Родительская таблица**

CREATE TABLE **ParentClass** (id INT AUTO\_INCREMENT PRIMARY KEY, class\_name VARCHAR(50) NOT NULL);

**Таблица с Домашними животными**

CREATE TABLE **HomeAnimals** (id INT AUTO\_INCREMENT PRIMARY KEY, parent\_id INT, name VARCHAR(50) NOT NULL, FOREIGN KEY (parent\_id) REFERENCES **ParentClass**(id));

**Таблица с Вьючными животными**

CREATE TABLE **PackAnimals** (id INT AUTO\_INCREMENT PRIMARY KEY, parent\_id INT, name VARCHAR(50) NOT NULL, FOREIGN KEY (parent\_id) REFERENCES **ParentClass**(id));

**Таблицы для подклассов домашних животных**

CREATE TABLE **Dogs** (id INT AUTO\_INCREMENT PRIMARY KEY, parent\_id INT, name VARCHAR(50) NOT NULL, commands VARCHAR(100), birth\_date DATE, FOREIGN KEY (parent\_id) REFERENCES **HomeAnimals**(id));

CREATE TABLE **Cats** (id INT AUTO\_INCREMENT PRIMARY KEY, parent\_id INT, name VARCHAR(50) NOT NULL, commands VARCHAR(100), birth\_date DATE, FOREIGN KEY (parent\_id) REFERENCES **HomeAnimals**(id));

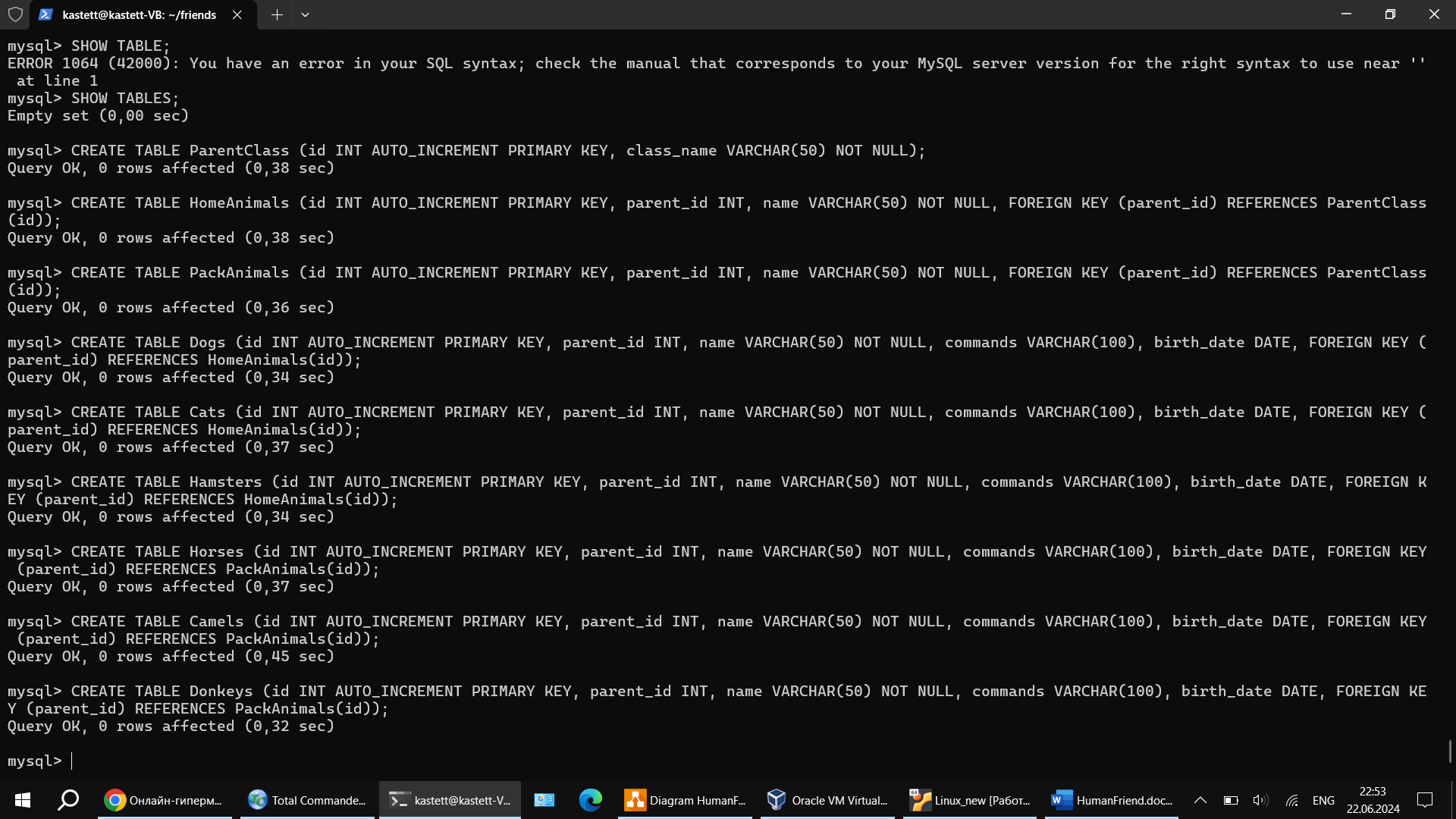
CREATE TABLE **Hamsters** (id INT AUTO\_INCREMENT PRIMARY KEY, parent\_id INT, name VARCHAR(50) NOT NULL, commands VARCHAR(100), birth\_date DATE, FOREIGN KEY (parent\_id) REFERENCES **HomeAnimals**(id));

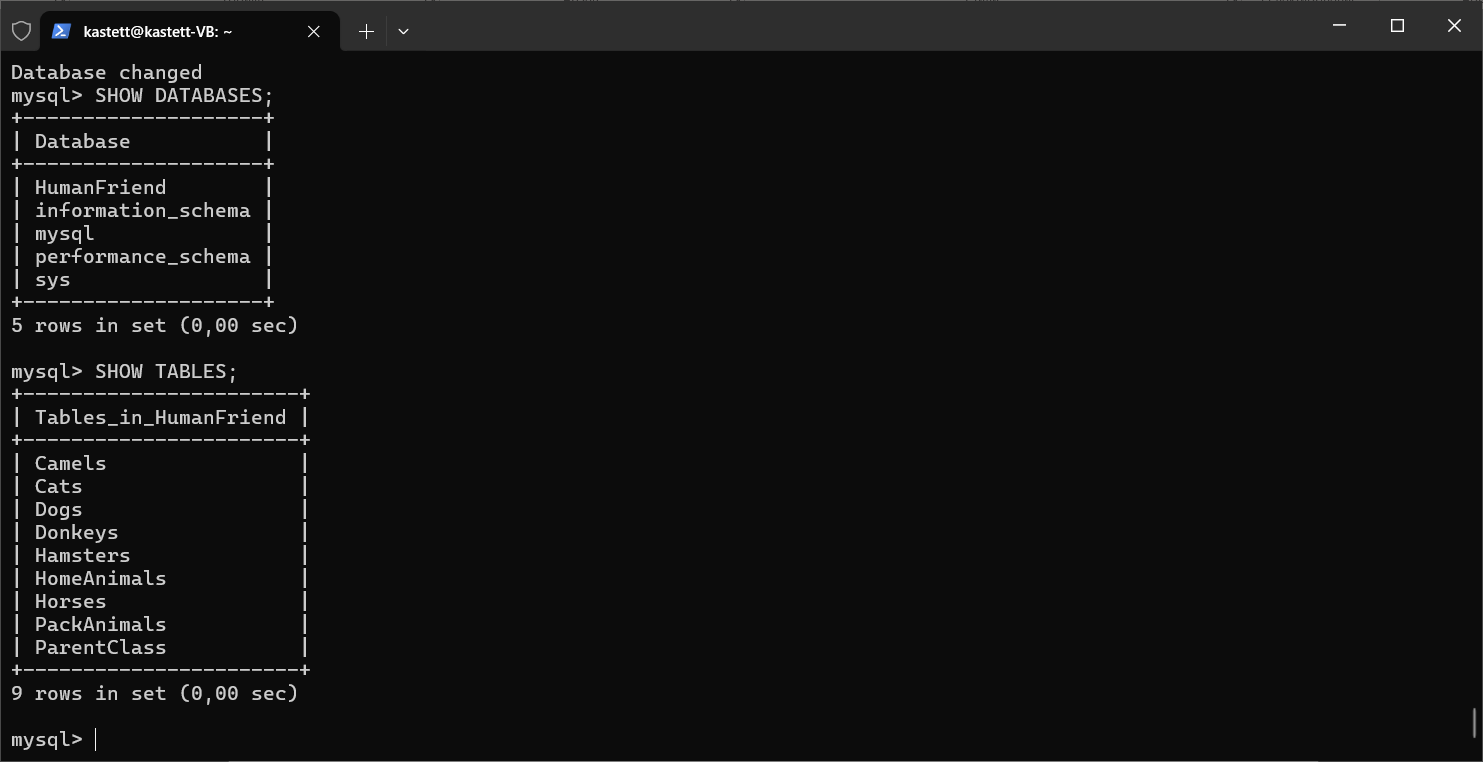
**Таблицы для подклассов вьючных животных**

CREATE TABLE **Horses** (id INT AUTO\_INCREMENT PRIMARY KEY, parent\_id INT, name VARCHAR(50) NOT NULL, commands VARCHAR(100), birth\_date DATE, FOREIGN KEY (parent\_id) REFERENCES **PackAnimals**(id));

CREATE TABLE **Camels** (id INT AUTO\_INCREMENT PRIMARY KEY, parent\_id INT, name VARCHAR(50) NOT NULL, commands VARCHAR(100), birth\_date DATE, FOREIGN KEY (parent\_id) REFERENCES **PackAnimals**(id));

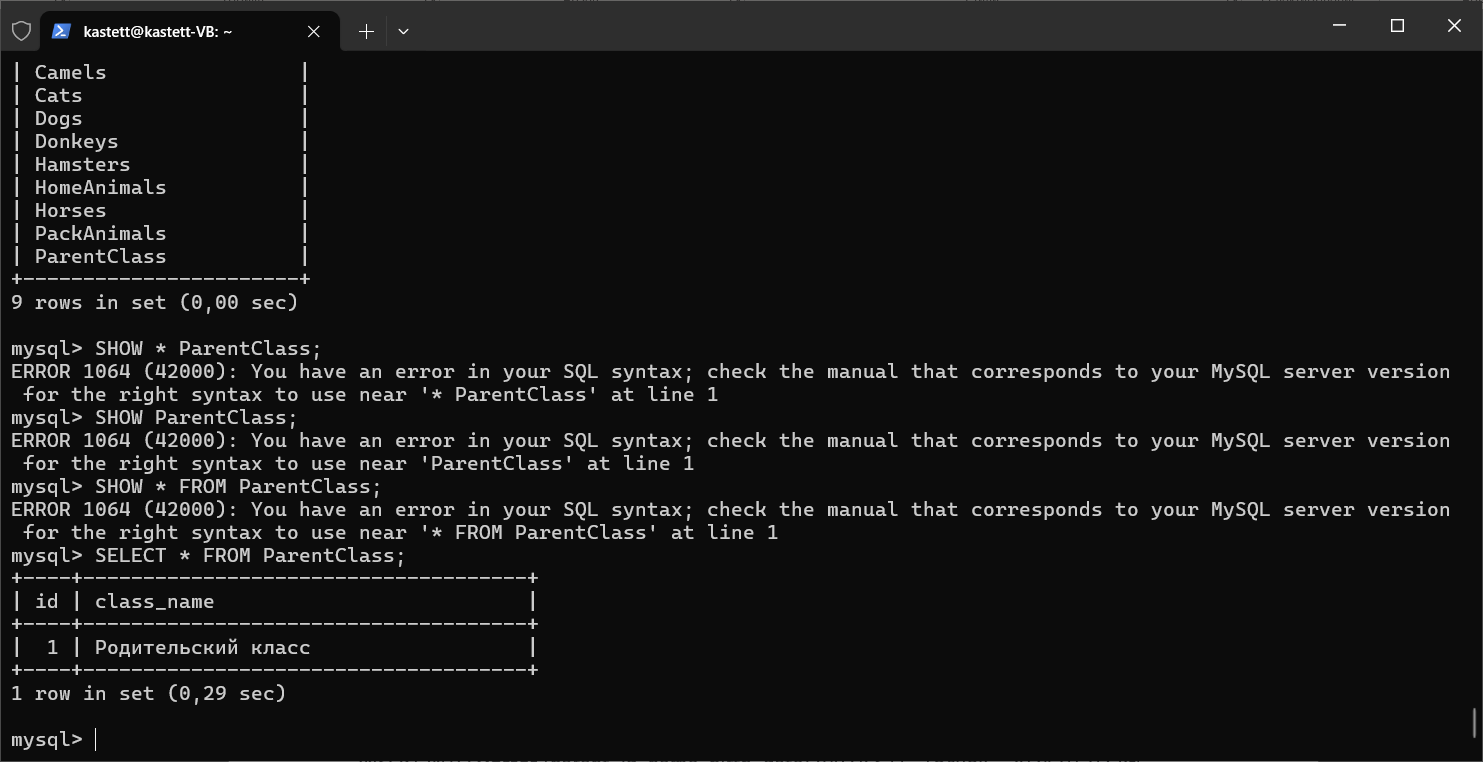
CREATE TABLE **Donkeys** (id INT AUTO\_INCREMENT PRIMARY KEY, parent\_id INT, name VARCHAR(50) NOT NULL, commands VARCHAR(100), birth\_date DATE, FOREIGN KEY (parent\_id) REFERENCES **PackAnimals**(id));





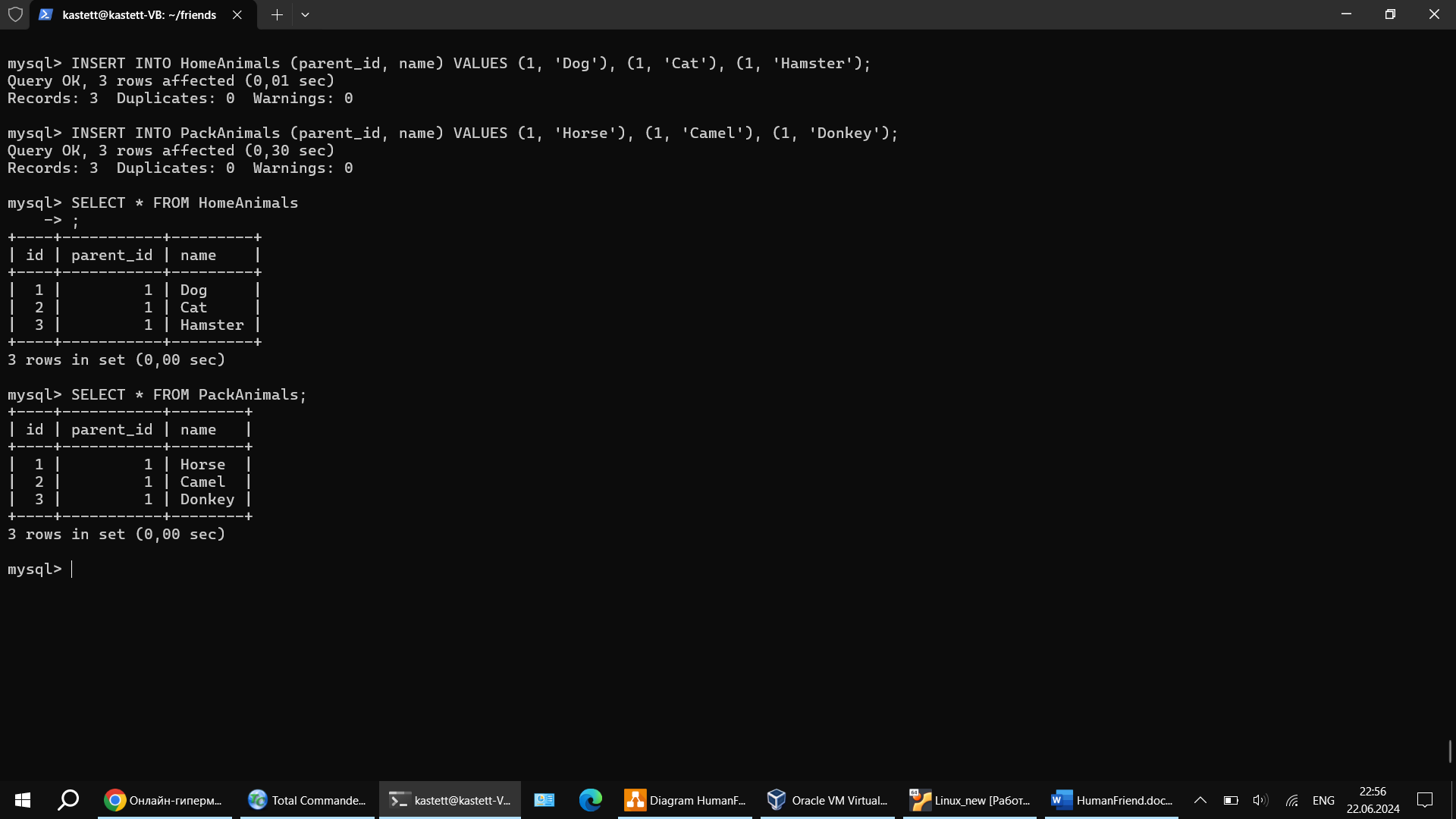
9. Заполнить низкоуровневые таблицы именами(животных), командами которые они выполняют и датами рождения.

INSERT INTO **ParentClass** (class\_name) VALUES ('Родительский класс');



INSERT INTO **HomeAnimals** (parent\_id, name) VALUES (1, 'Dog'), (1, 'Cat'), (1, 'Hamster');

INSERT INTO **PackAnimals** (parent\_id, name) VALUES (1, 'Horse'), (1, 'Camel'), (1, 'Donkey');



INSERT INTO **Dogs** (parent\_id, name, commands, birth\_date) VALUES (1, 'Sharik', 'Сидеть, Лежать, Дать лапу', '2024-01-01');

INSERT INTO **Dogs** (parent\_id, name, commands, birth\_date) VALUES (1, 'Bobik', 'Сидеть, Лежать, Дать лапу', '2022-01-02');

INSERT INTO **Cats** (parent\_id, name, commands, birth\_date) VALUES (2, 'Tom', 'Муркать, Играть', '2024-02-01');

INSERT INTO **Cats** (parent\_id, name, commands, birth\_date) VALUES (2, 'Masya', 'Муркать, Играть', '2022-02-02');

INSERT INTO **Hamsters** (parent\_id, name, commands, birth\_date) VALUES (3, 'Mike', 'Бегать в колесе, Собирать еду', '2024-03-01');

INSERT INTO **Hamsters** (parent\_id, name, commands, birth\_date) VALUES (3, 'Nike', 'Бегать в колесе, Собирать еду', '2022-03-02');

INSERT INTO **Horses** (parent\_id, name, commands, birth\_date) VALUES (1, 'Gvozd', 'Тянуть повозку, Возить всадника', '2024-01-01');

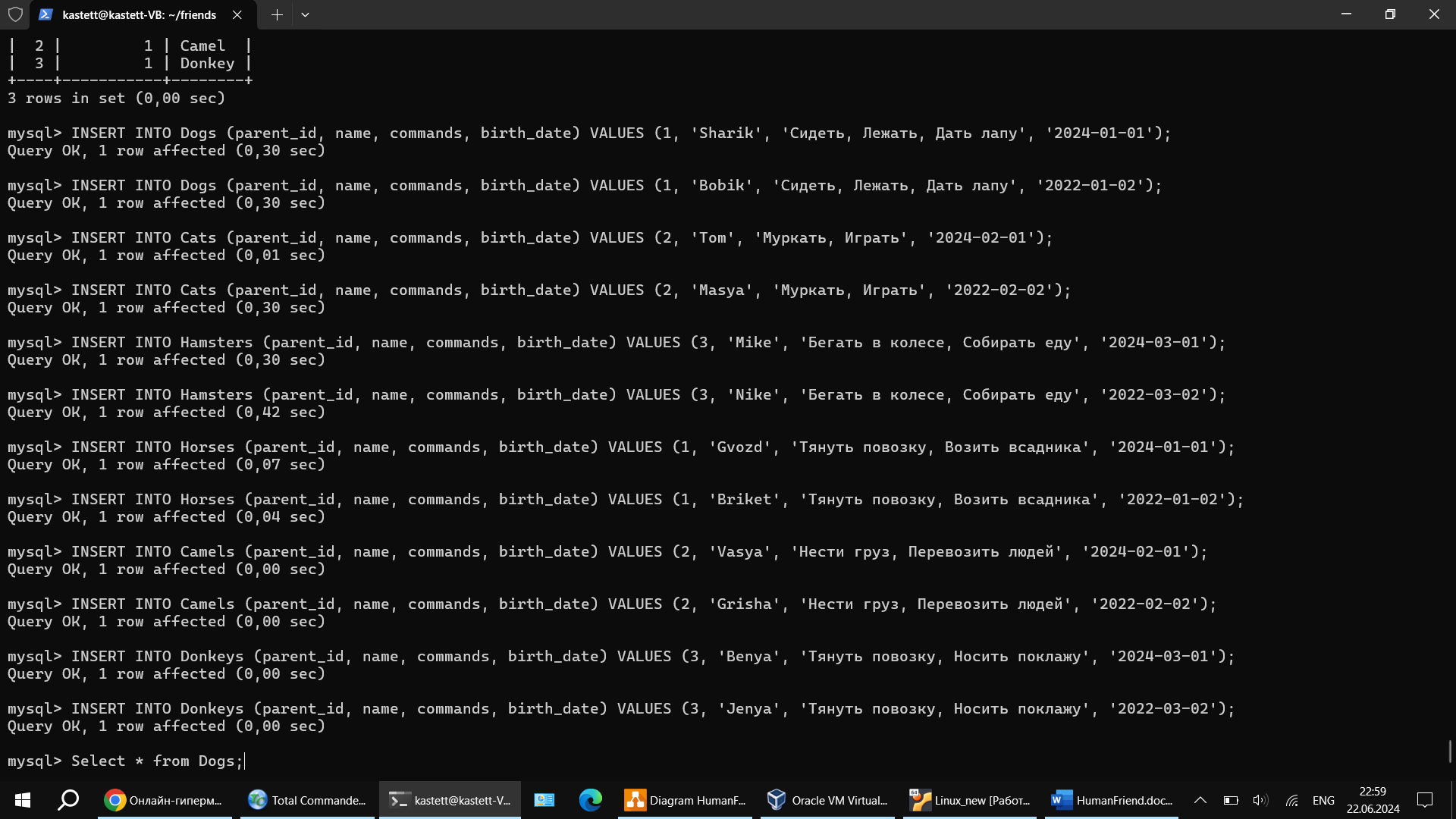
INSERT INTO **Horses** (parent\_id, name, commands, birth\_date) VALUES (1, 'Briket', 'Тянуть повозку, Возить всадника', '2022-01-02');

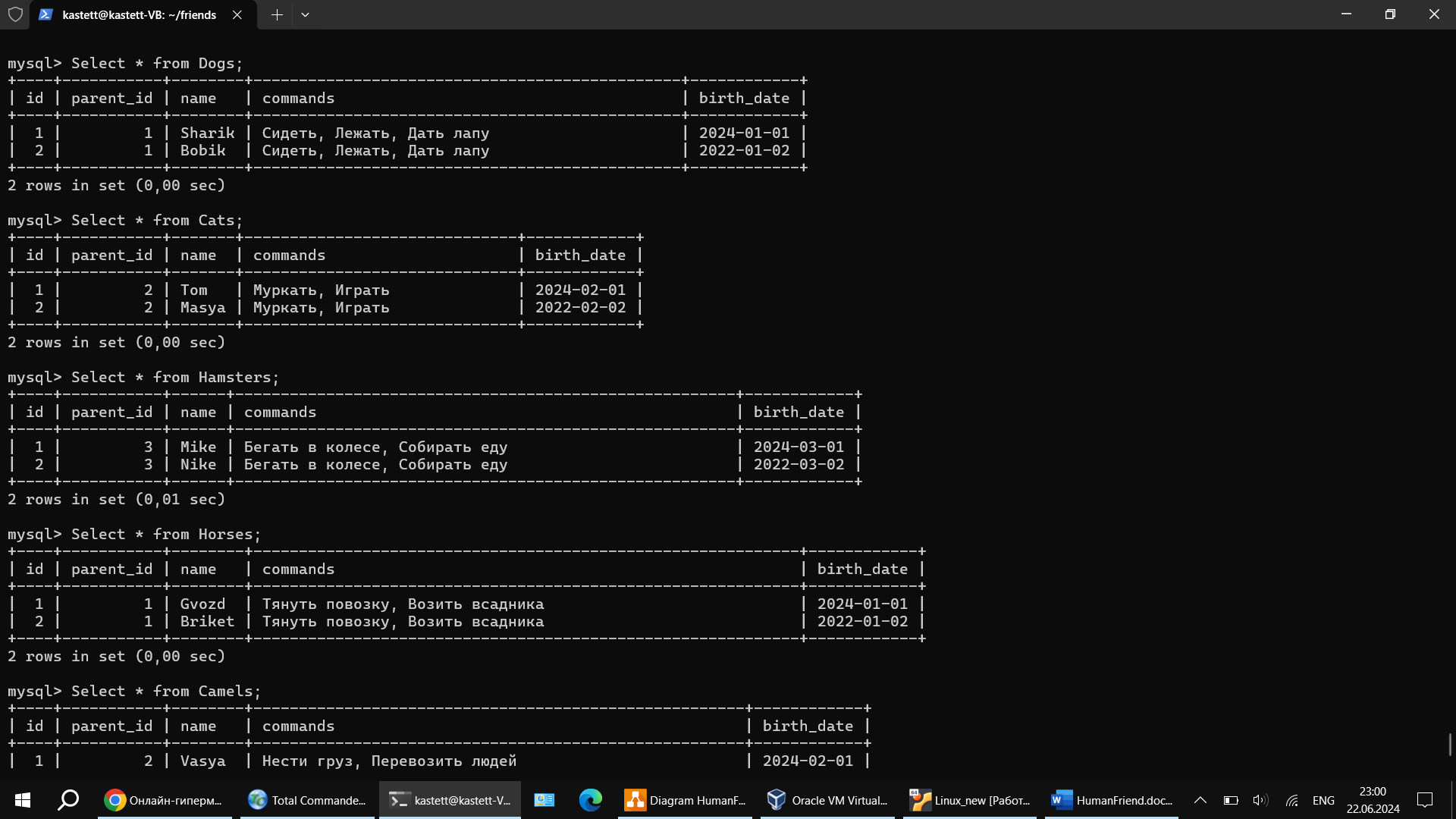
INSERT INTO **Camels** (parent\_id, name, commands, birth\_date) VALUES (2, 'Vasya', 'Нести груз, Перевозить людей', '2024-02-01');

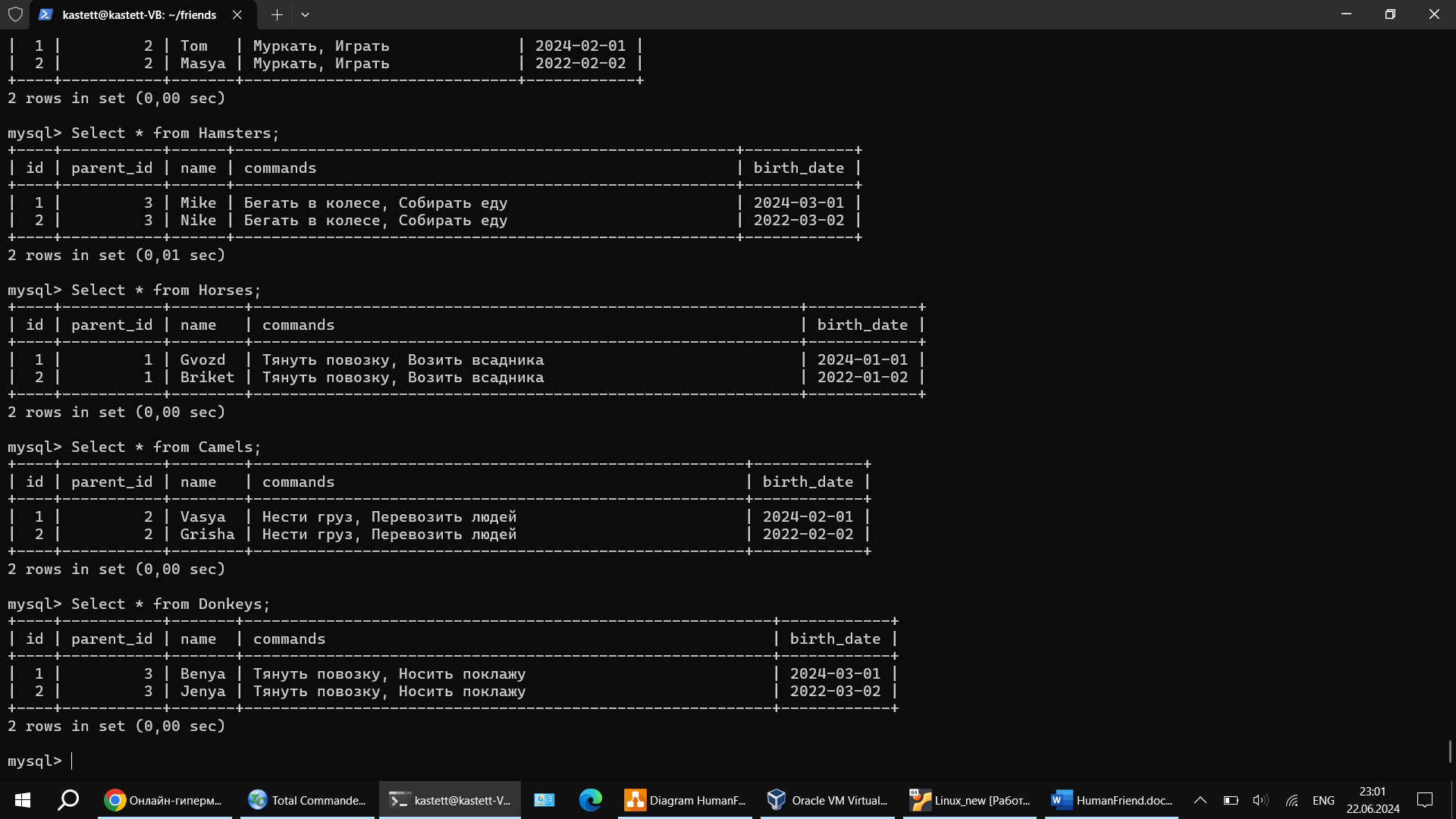
INSERT INTO **Camels** (parent\_id, name, commands, birth\_date) VALUES (2, 'Grisha', 'Нести груз, Перевозить людей', '2022-02-02');

INSERT INTO **Donkeys** (parent\_id, name, commands, birth\_date) VALUES (3, 'Benya', 'Тянуть повозку, Носить поклажу', '2024-03-01');

INSERT INTO **Donkeys** (parent\_id, name, commands, birth\_date) VALUES (3, 'Jenya', 'Тянуть повозку, Носить поклажу', '2022-03-02');



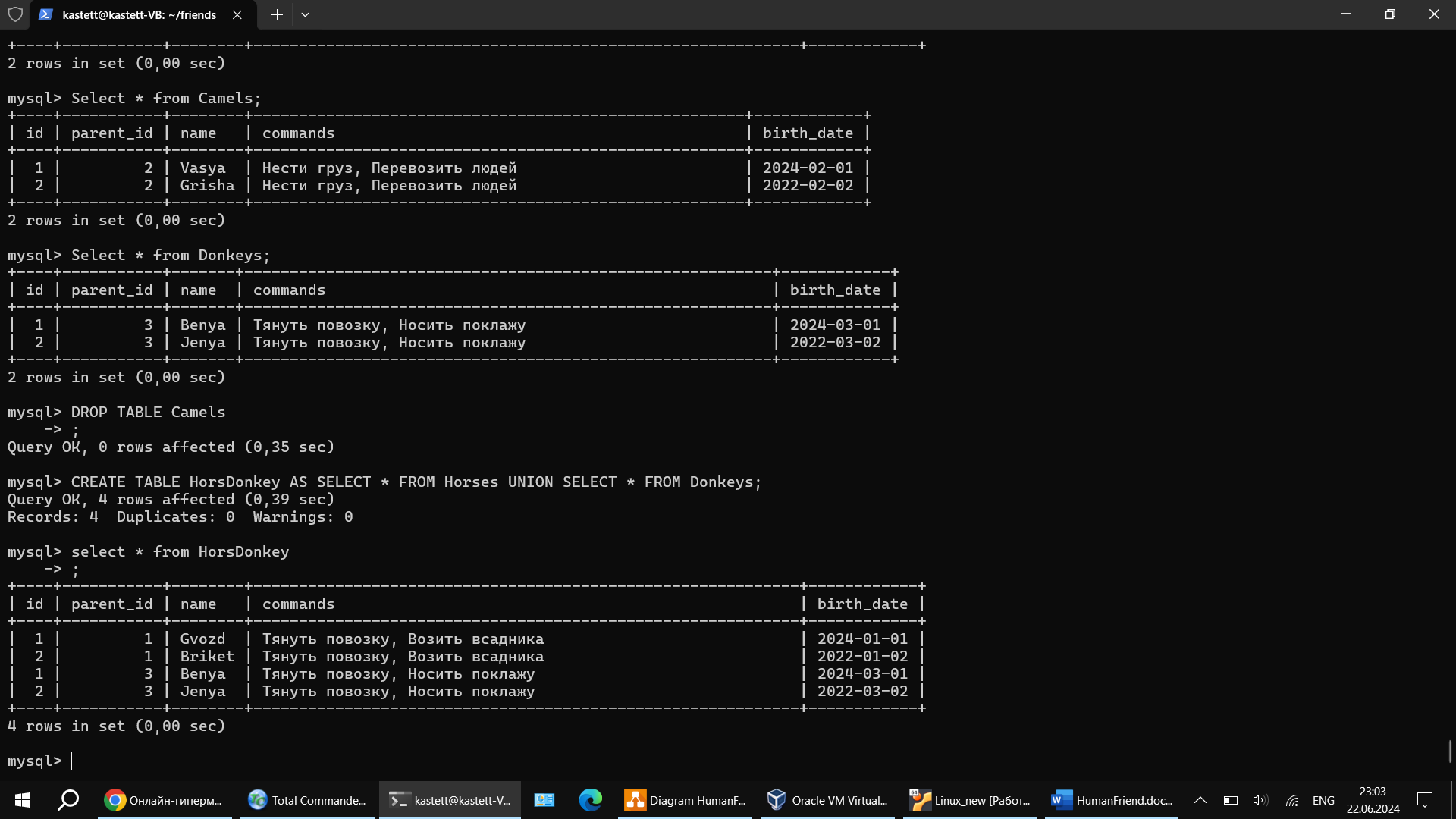




10. Удалив из таблицы верблюдов, т.к. верблюдов решили перевезти в другой питомник на зимовку. Объединить таблицы лошади, и ослы в одну таблицу. (ОК)

DROP TABLE Camels

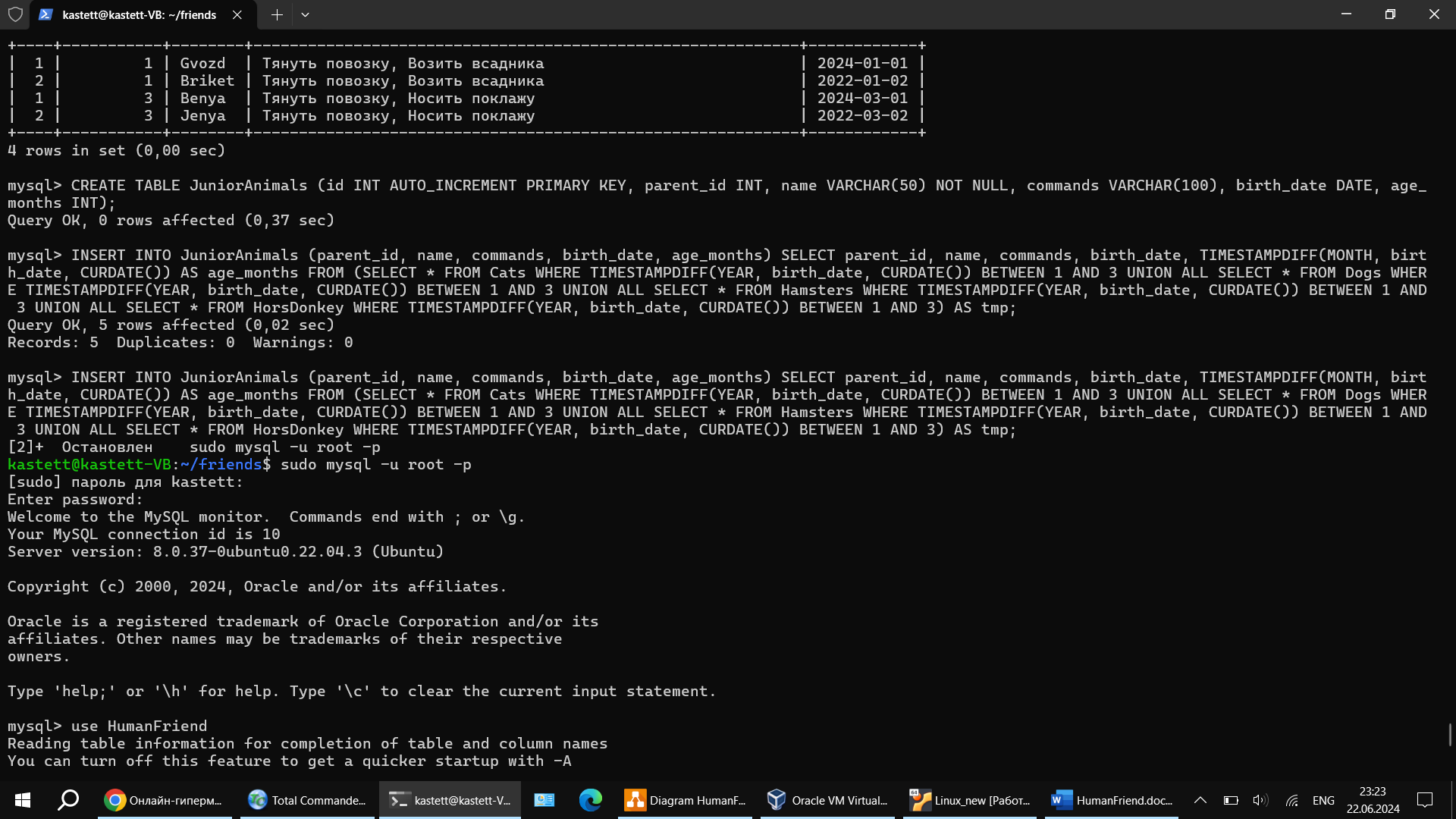
CREATE TABLE HorsDonkey AS SELECT \* FROM Horses UNION SELECT \* FROM Donkeys;

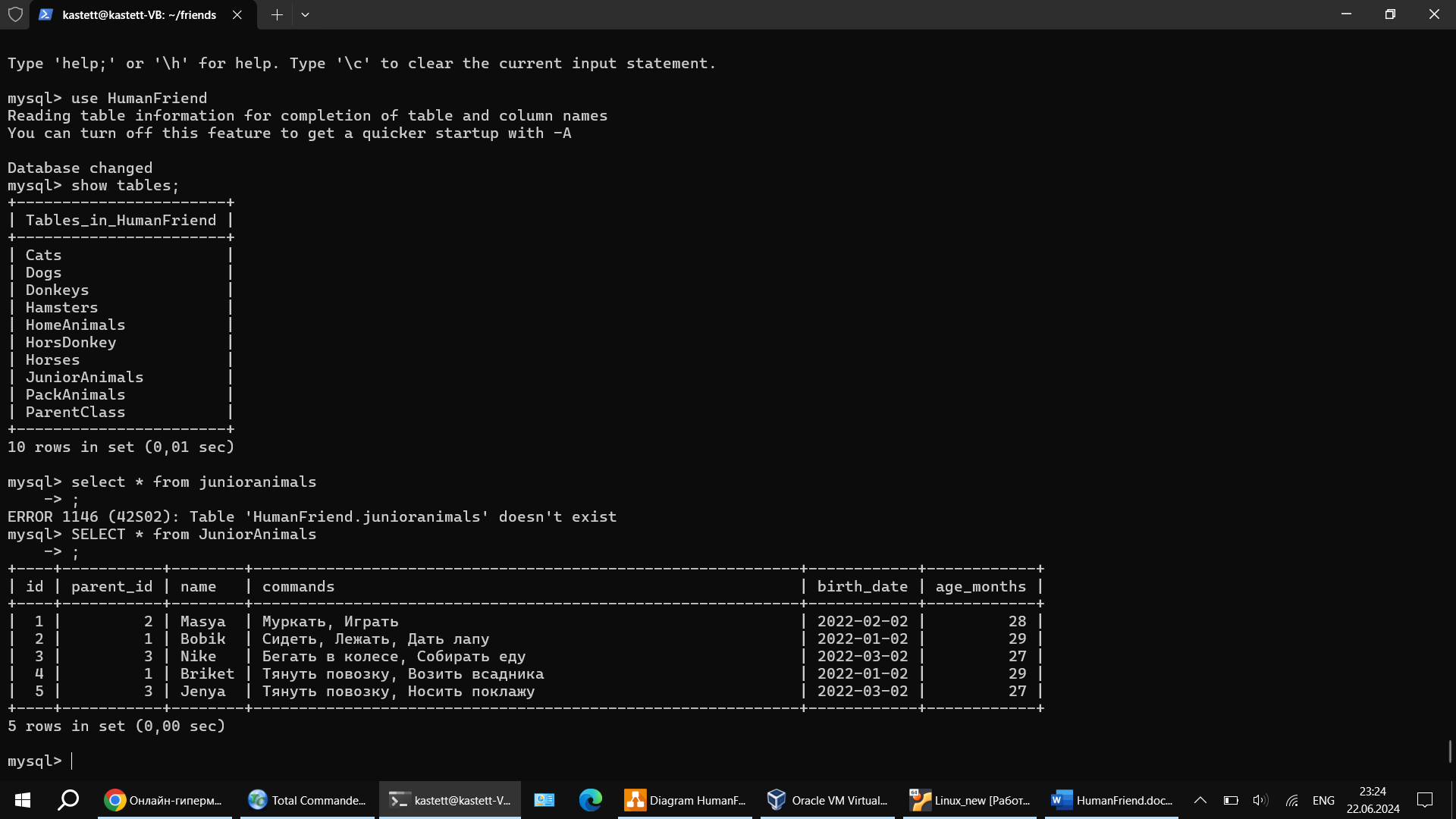


11.Создать новую таблицу “молодые животные” (JuniorAnimals) в которую попадут все животные **старше 1 года**, но **младше 3 лет** и в отдельном столбце с **точностью до месяца подсчитать возраст** животных в новой таблице. (OK)

CREATE TABLE **JuniorAnimals** (id INT AUTO\_INCREMENT PRIMARY KEY, parent\_id INT, name VARCHAR(50) NOT NULL, commands VARCHAR(100), birth\_date DATE, age\_months INT);

INSERT INTO **JuniorAnimals** (parent\_id, name, commands, birth\_date, age\_months) SELECT parent\_id, name, commands, birth\_date, TIMESTAMPDIFF(MONTH, birth\_date, CURDATE()) AS age\_months FROM (SELECT \* FROM Cats WHERE TIMESTAMPDIFF(YEAR, birth\_date, CURDATE()) BETWEEN 1 AND 3 UNION ALL SELECT \* FROM Dogs WHERE TIMESTAMPDIFF(YEAR, birth\_date, CURDATE()) BETWEEN 1 AND 3 UNION ALL SELECT \* FROM Hamsters WHERE TIMESTAMPDIFF(YEAR, birth\_date, CURDATE()) BETWEEN 1 AND 3 UNION ALL SELECT \* FROM HorsDonkey WHERE TIMESTAMPDIFF(YEAR, birth\_date, CURDATE()) BETWEEN 1 AND 3) AS tmp;

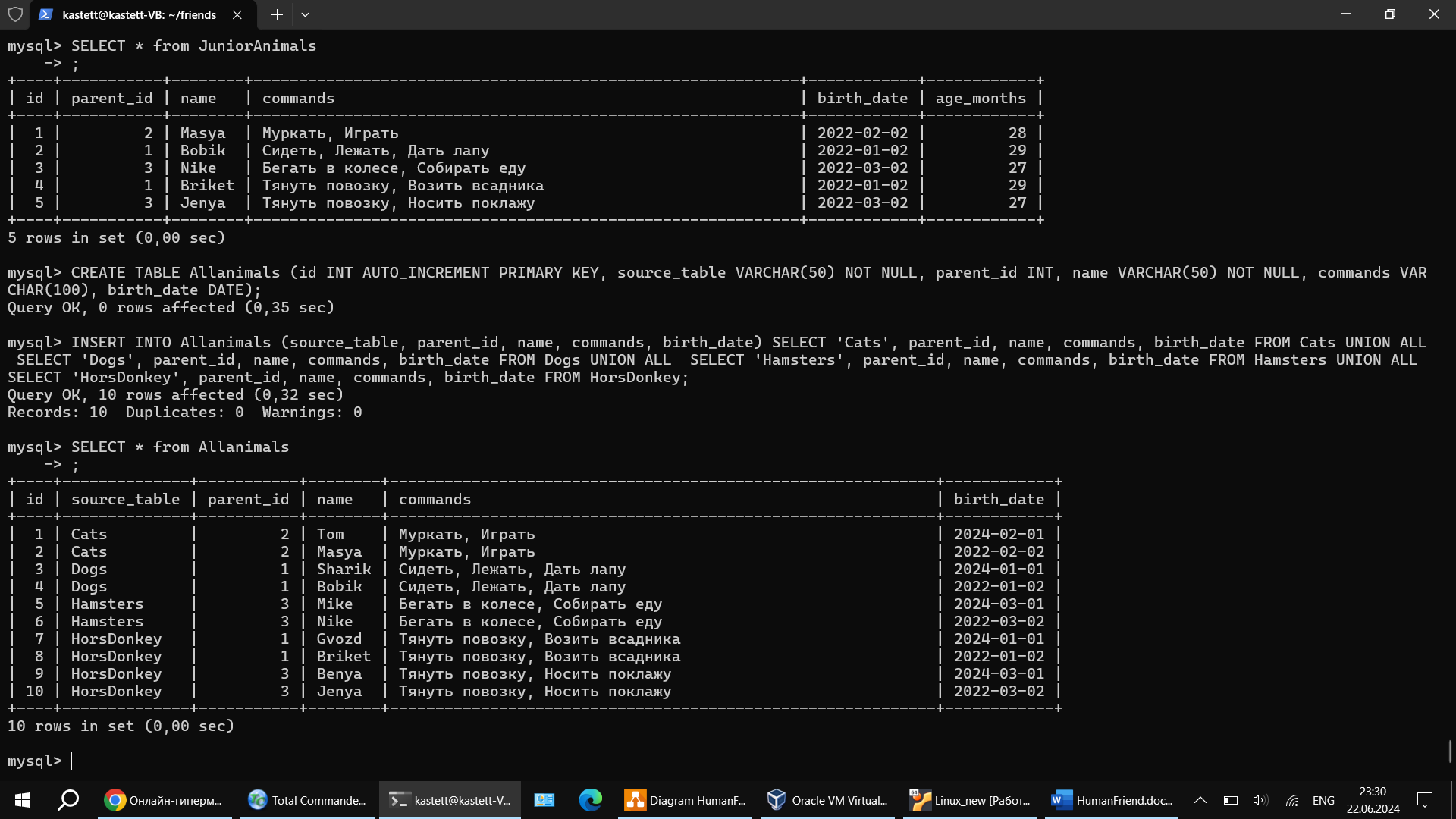




12. Объединить все таблицы в одну, при этом сохраняя поля, указывающие на прошлую принадлежность к старым таблицам.

CREATE TABLE **Allanimals** (id INT AUTO\_INCREMENT PRIMARY KEY, source\_table VARCHAR(50) NOT NULL, parent\_id INT, name VARCHAR(50) NOT NULL, commands VARCHAR(100), birth\_date DATE);

INSERT INTO **Allanimals** (source\_table, parent\_id, name, commands, birth\_date) SELECT 'Cats', parent\_id, name, commands, birth\_date FROM Cats UNION ALL SELECT 'Dogs', parent\_id, name, commands, birth\_date FROM Dogs UNION ALL SELECT 'Hamsters', parent\_id, name, commands, birth\_date FROM Hamsters UNION ALL SELECT 'HorsDonkey', parent\_id, name, commands, birth\_date FROM HorsDonkey;



13.Создать класс с Инкапсуляцией методов и наследованием по диаграмме.(OK)

14. Написать программу, имитирующую работу реестра домашних животных. (ОК)

В программе должен быть реализован следующий функционал:

14.1 Завести новое животное

14.2 определять животное в правильный класс

14.3 увидеть список команд, которое выполняет животное

14.4 обучить животное новым командам

14.5 Реализовать навигацию по меню

15. Выполнить следующие модификации программы:

15.1. Создать **класс Счетчик**, у которого есть метод **add()**, увеличивающий̆ значение внутренней̆ int переменной̆ на 1 при нажатие “**Завести новое животное**”.

15.2. Сделать так, чтобы с объектом такого типа (**класса Счетчик**) можно было работать в блоке **try-with-resources**.

15.3. Нужно бросить исключение, если работа с объектом (**типа Счетчик**) была не в ресурсном **try** и/или ресурс **остался открыт**.

15.4. Значение считать в ресурсе **try**, если при заведения животного **заполнены все поля**.