

**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

|  |
| --- |
| **РТУ МИРЭА** |
|  |
| **Институт кибербезопасности и цифровых технологий (ИКБ)** |
|  |
| КБ-2 «Информационно-аналитические системы кибербезопасности» |

**ОТЧЕТ О ВЫПОЛНЕНИИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ №6**

# **В РАМКАХ ДИСЦИПЛИНЫ «ТЕХНОЛОГИИ ХРАНЕНИЯ В СИСТЕМАХ КИБЕРБЕЗОПАСНОСТИ»**

Выполнил:

Студент 3-ого курса

Учебной группы БИСО-02-22

Зубарев В.С.

Создайте каталог для нового проекта (например, spark) и сформируйте файл docker-compose.yml для развертывания контейнера c Jupyter Notebook и PySpark (<https://quay.io/repository/jupyter/pyspark-notebook>).

version: '3.8'

services:

pyspark:

image: quay.io/jupyter/pyspark-notebook:latest

container\_name: pyspark\_zvs

volumes:

- ./notebooks:/home/jovyan/work

ports:

- 8805:8888

- 4005:4040

- 4105:4041

networks:

- pyspark-net

environment:

- JUPYTER\_ENABLE\_LAB=yes

command: start-notebook.sh --NotebookApp.token='' NotebookApp.password='' --NotebookApp.allow\_origin='\*' --NotebookApp.disable\_check\_xsrf=True

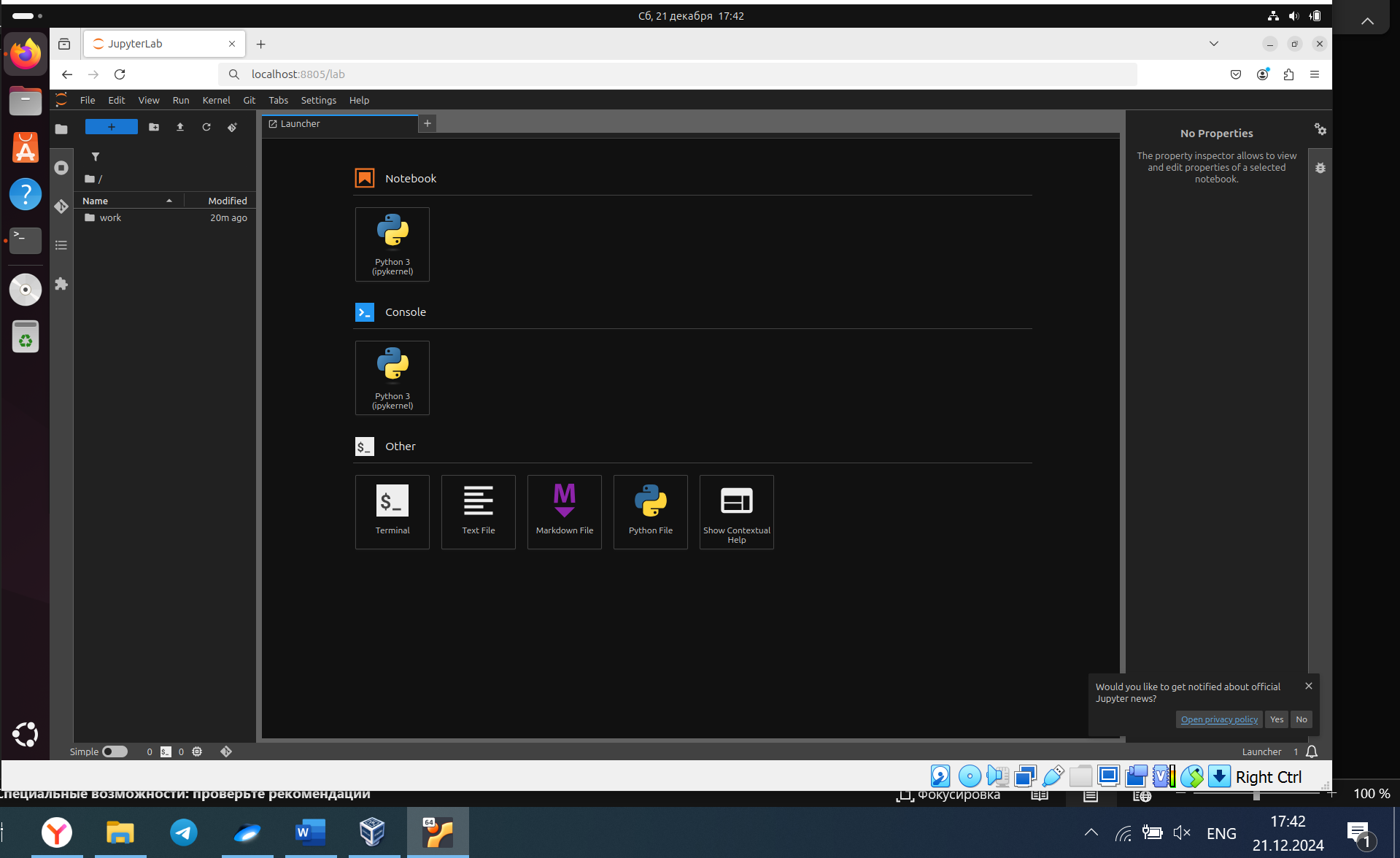
networks:

pyspark-net:

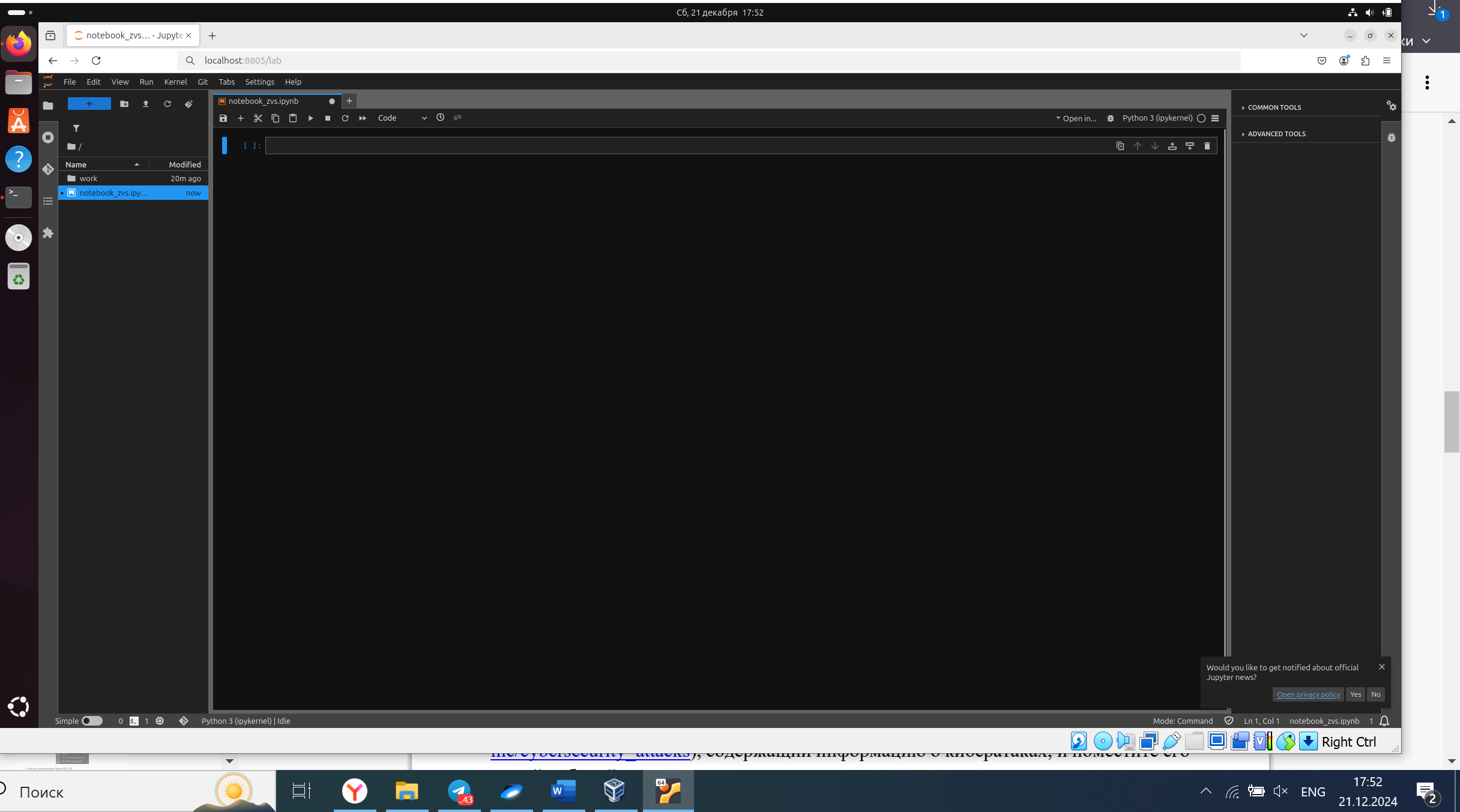
driver: bridge



Разверните контейнер с помощью Docker Compose



Создайте блокнот (notebook). Название блокнота должно заканчиваться на символ подчеркивания и инициалы ФИО.



Загрузите CSV-файл cybersecurity\_attacks.csv (https://github.com/incriboinc/cybersecurity\_attacks), содержащий информацию о кибератаках, и поместите его в свой рабочий каталог.



Импортируйте модули библиотеки PySpark, содержащие функции и типы данных для работы с данными в Spark DataFrame (pyspark.sql.functions, pyspark.sql.types). Создайте сессию Spark. Название приложения должно заканчиваться на символ подчеркивания и инициалы ФИО.

from pyspark.sql import SparkSession

import pyspark.sql.functions as f

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType, DateType

# Создание сессии Spark

spark = SparkSession.builder.appName("spark\_zvs").getOrCreate()

# Загрузка данных из CSV файла

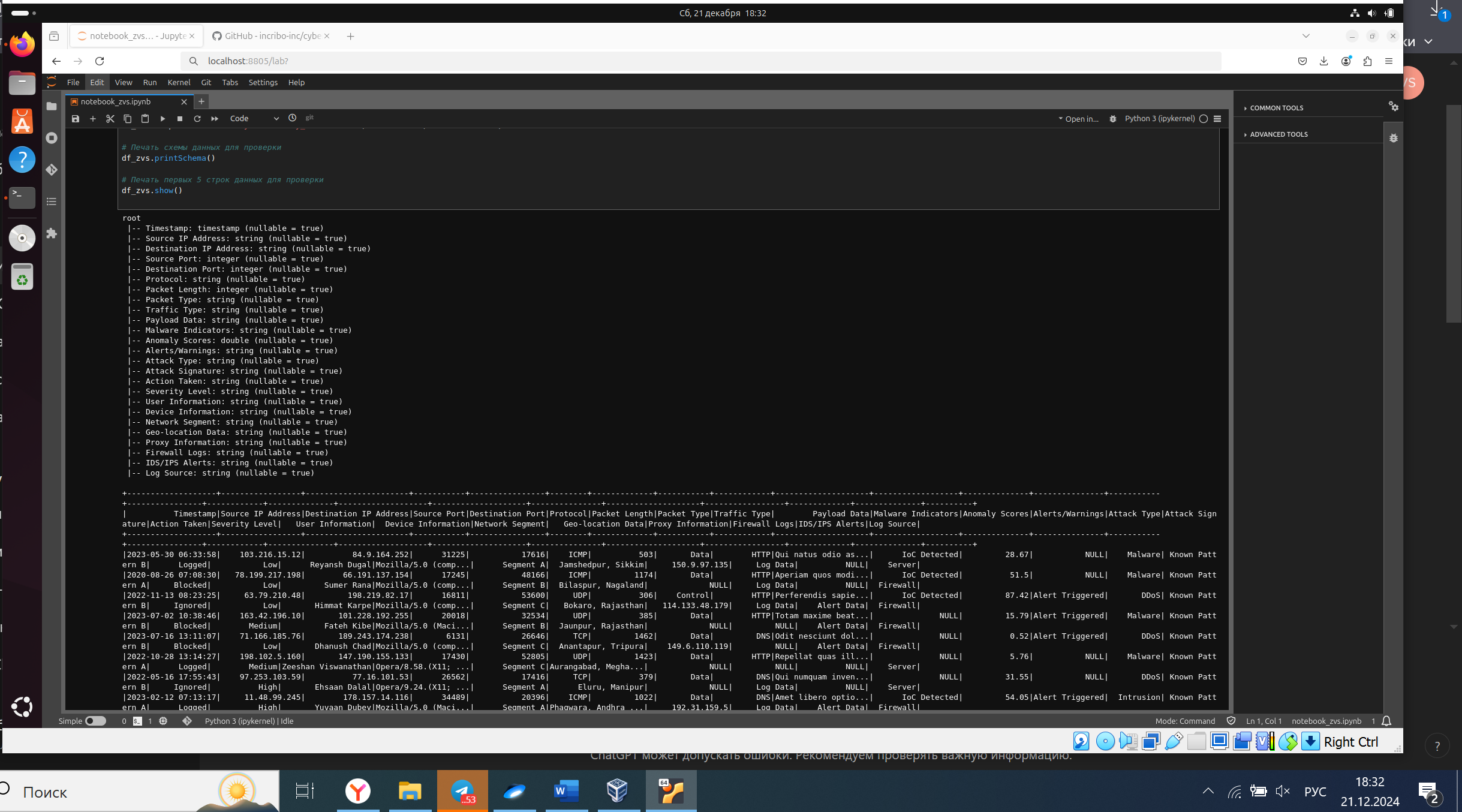
df\_zvs = spark.read.csv("cybersecurity\_attacks.csv", header=True, multiLine=True, inferSchema=True)

# Печать схемы данных для проверки

df\_zvs.printSchema()

# Печать первых 5 строк данных для проверки

df\_zvs.show()



Сделать выборку необходимых полей можно с помощью функции select.

from pyspark.sql import SparkSession

import pyspark.sql.functions as f

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType, DateType

# Создание сессии Spark

spark = SparkSession.builder.appName("spark\_zvs").getOrCreate()

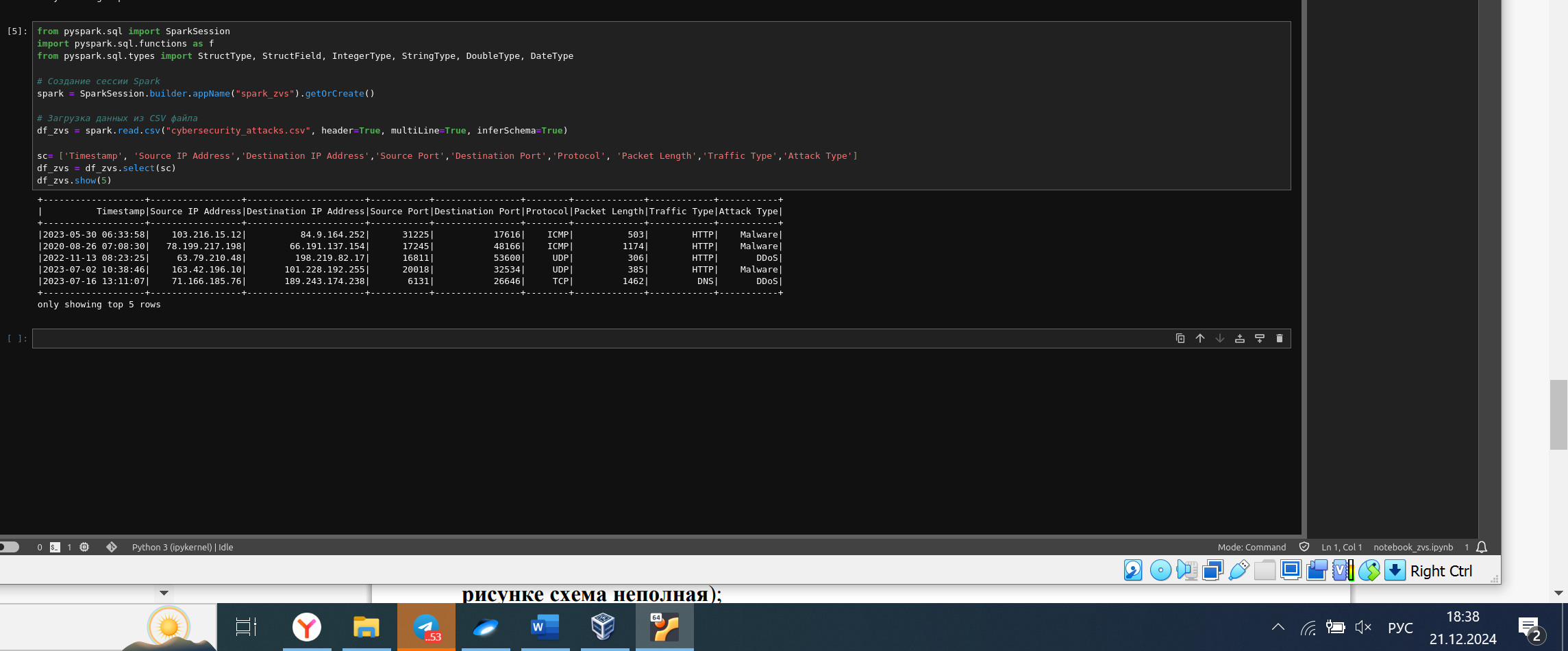
# Загрузка данных из CSV файла

df\_zvs = spark.read.csv("cybersecurity\_attacks.csv", header=True, multiLine=True, inferSchema=True)

sc= ['Timestamp', 'Source IP Address','Destination IP Address','Source Port','Destination Port','Protocol', 'Packet Length','Traffic Type','Attack Type']

df\_zvs = df\_zvs.select(sc)

df\_zvs.show(5)



Приведите поля к соответствующим типам данных, предварительно создав схему данных и использовав ее при загрузке

from pyspark.sql import SparkSession

import pyspark.sql.functions as f

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType, DateType

# Создание сессии Spark

spark = SparkSession.builder.appName("spark\_zvs").getOrCreate()

# Определение схемы для загрузки данных

schema = StructType([

StructField("Timestamp", DateType(), True),

StructField("Source IP Address", StringType(), True),

StructField("Destination IP Address", StringType(), True),

StructField("Source Port", IntegerType(), True),

StructField("Destination Port", IntegerType(), True),

StructField("Protocol", StringType(), True),

StructField("Packet Length", IntegerType(), True),

StructField("Packet Type", StringType(), True),

StructField("Traffic Type", StringType(), True),

StructField("Payload Data", StringType(), True),

StructField("Malware Indicators", StringType(), True),

StructField("Anomaly Scores", DoubleType(), True),

StructField("Alerts/Warnings", StringType(), True),

StructField("Attack Type", StringType(), True),

StructField("Attack Signature", StringType(), True),

StructField("Action Taken", StringType(), True),

StructField("Severity Level", StringType(), True),

StructField("User Information", StringType(), True),

StructField("Device Information", StringType(), True),

StructField("Network Segment", StringType(), True),

StructField("Geo-location Data", StringType(), True),

StructField("Proxy Information", StringType(), True),

StructField("Firewall Logs", StringType(), True),

StructField("IDS/IPS Alerts", StringType(), True),

StructField("Log Source", StringType(), True)

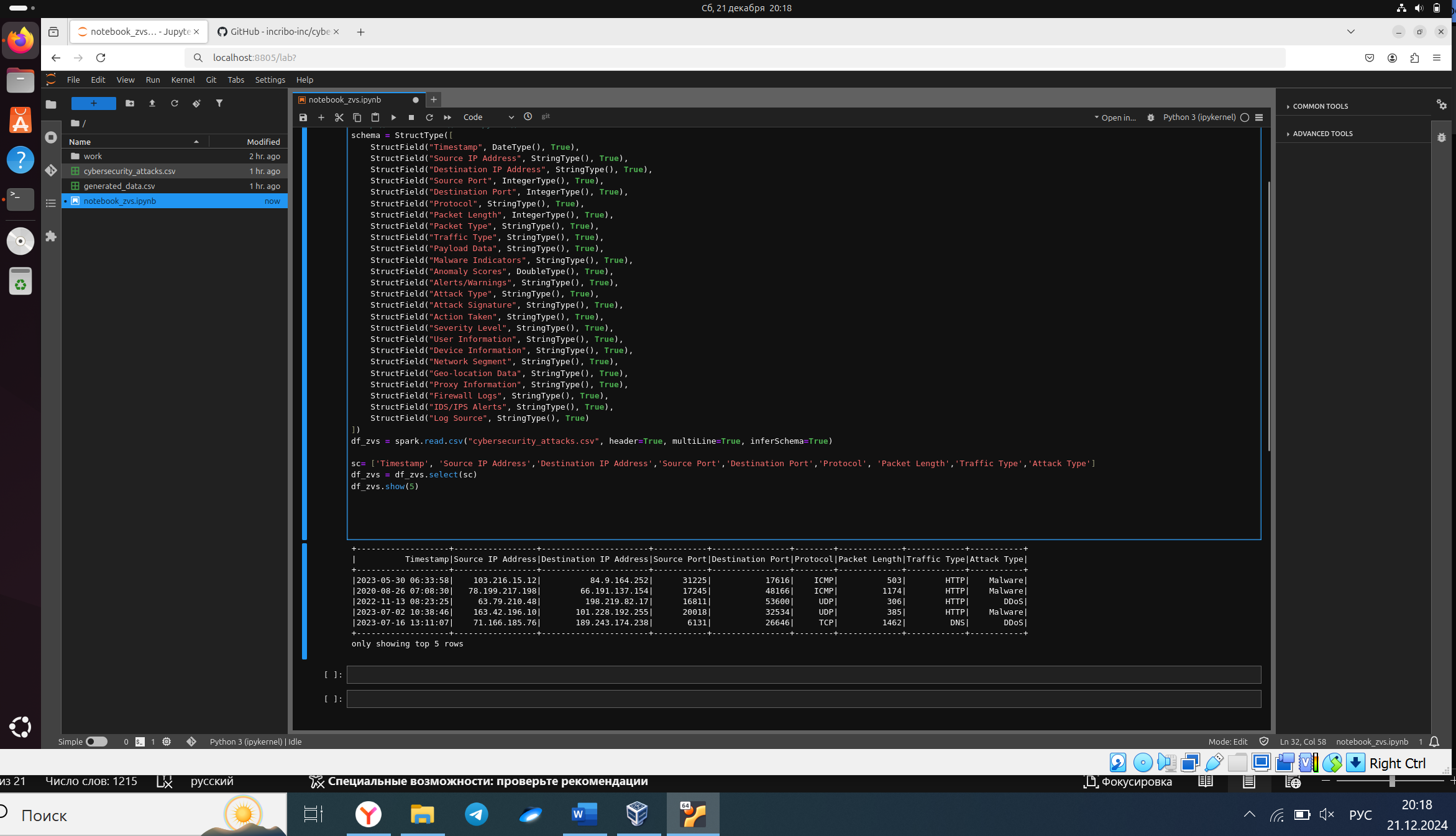
])

df\_zvs = spark.read.csv("cybersecurity\_attacks.csv", header=True, multiLine=True, inferSchema=True)

sc= ['Timestamp', 'Source IP Address','Destination IP Address','Source Port','Destination Port','Protocol', 'Packet Length','Traffic Type','Attack Type']

df\_zvs = df\_zvs.select(sc)

df\_zvs.show(5)



Выведите 10 записей, в которых порт источника (Source Port) делится без остатка на ваш номер по списку, умноженный на 10.

from pyspark.sql import SparkSession

import pyspark.sql.functions as f

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType, DateType

# Создание сессии Spark

spark = SparkSession.builder.appName("spark\_zvs").getOrCreate()

# Загрузка данных из CSV файла

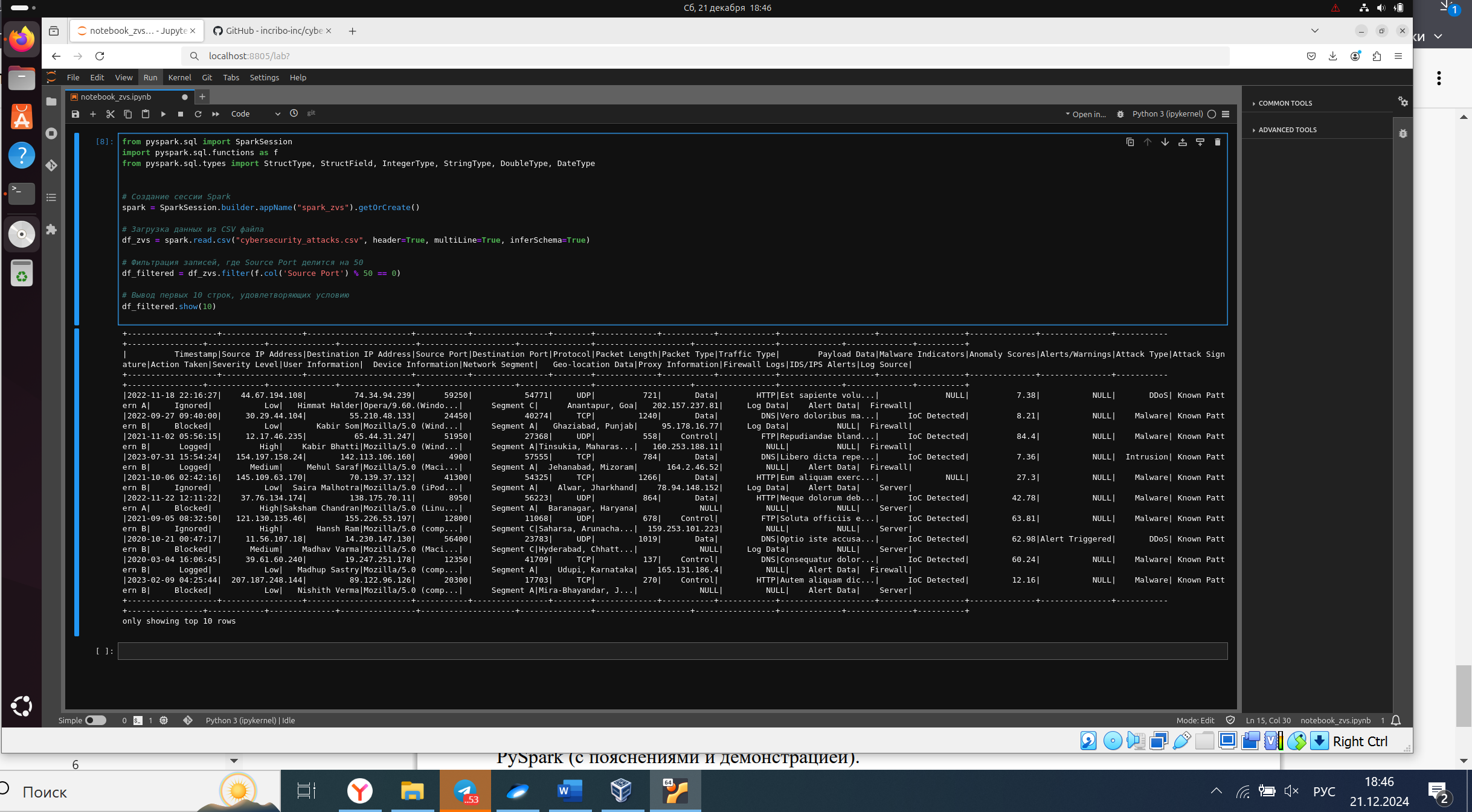
df\_zvs = spark.read.csv("cybersecurity\_attacks.csv", header=True, multiLine=True, inferSchema=True)

# Фильтрация записей, где Source Port делится на 50

df\_filtered = df\_zvs.filter(f.col('Source Port') % 50 == 0)

# Вывод первых 10 строк, удовлетворяющих условию

df\_filtered.show(10)



Выведите результат группировки по типу трафика (Traffic Type) с подсчетом количества записей.

from pyspark.sql import SparkSession

import pyspark.sql.functions as f

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType, DateType

# Создание сессии Spark

spark = SparkSession.builder.appName("spark\_zvs").getOrCreate()

# Загрузка данных из CSV файла

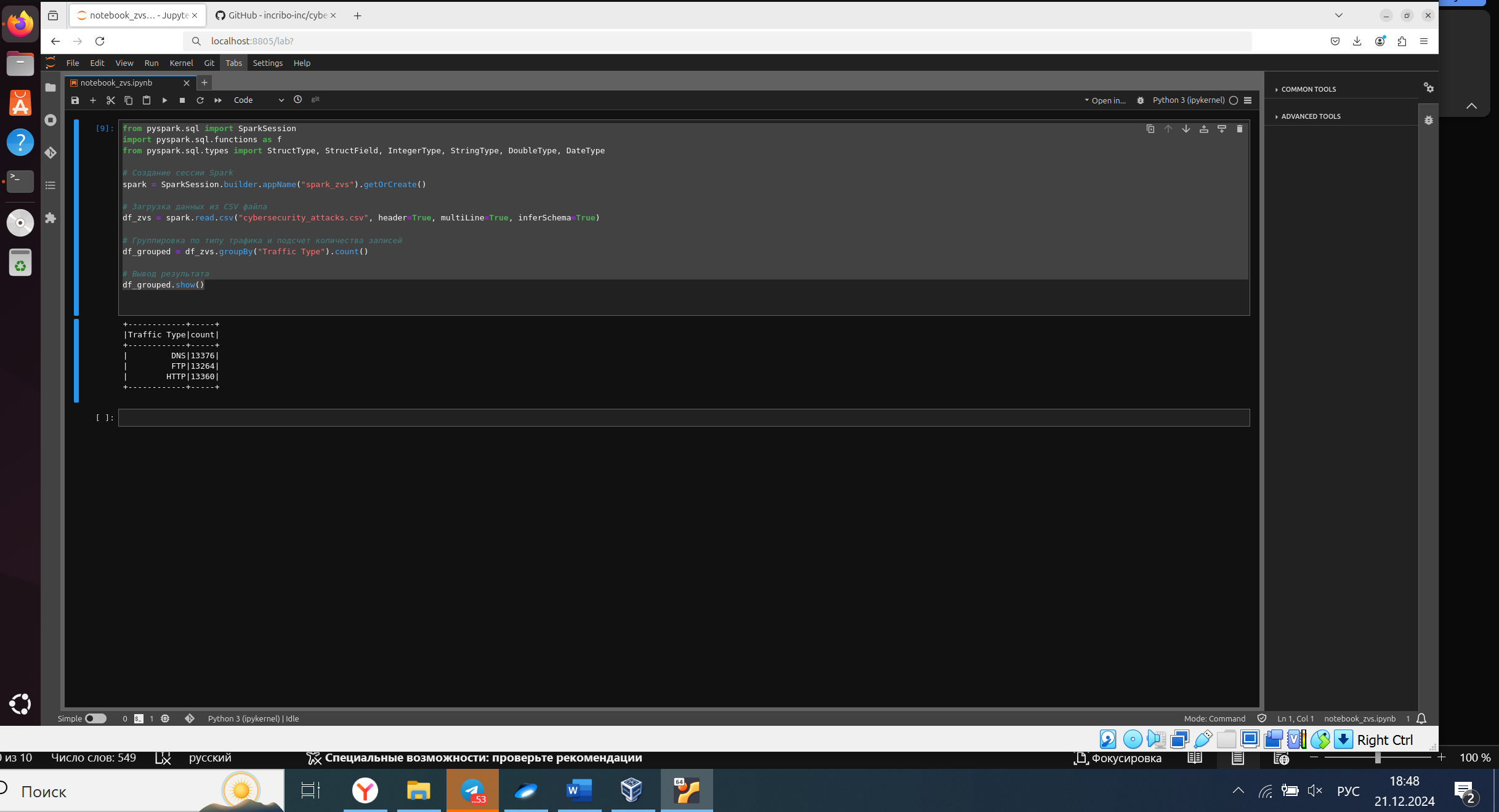
df\_zvs = spark.read.csv("cybersecurity\_attacks.csv", header=True, multiLine=True, inferSchema=True)

# Группировка по типу трафика и подсчет количества записей

df\_grouped = df\_zvs.groupBy("Traffic Type").count()

# Вывод результата

df\_grouped.show()



Выведите результат группировки по протоколу (Protocol) с подсчетом среднего размера пакета

from pyspark.sql import SparkSession

import pyspark.sql.functions as f

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType, DateType

# Создание сессии Spark

spark = SparkSession.builder.appName("spark\_zvs").getOrCreate()

# Загрузка данных из CSV файла

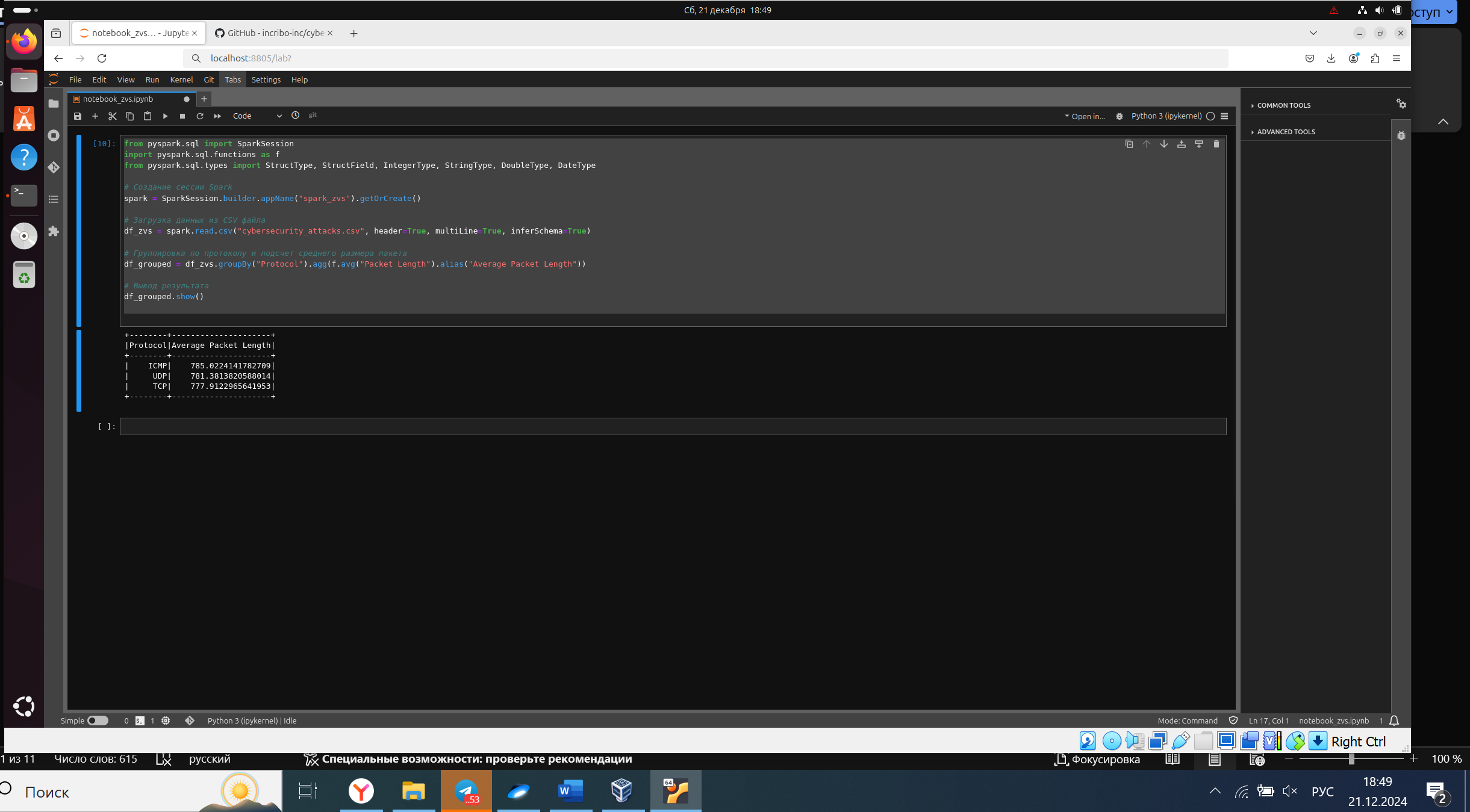
df\_zvs = spark.read.csv("cybersecurity\_attacks.csv", header=True, multiLine=True, inferSchema=True)

# Группировка по протоколу и подсчет среднего размера пакета

df\_grouped = df\_zvs.groupBy("Protocol").agg(f.avg("Packet Length").alias("Average Packet Length"))

# Вывод результата

df\_grouped.show()



Выведите 10 IP-адресов получателя, отсортированных в порядке убывания.

from pyspark.sql import SparkSession

import pyspark.sql.functions as f

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType, DateType

# Создание сессии Spark

spark = SparkSession.builder.appName("spark\_zvs").getOrCreate()

# Загрузка данных из CSV файла

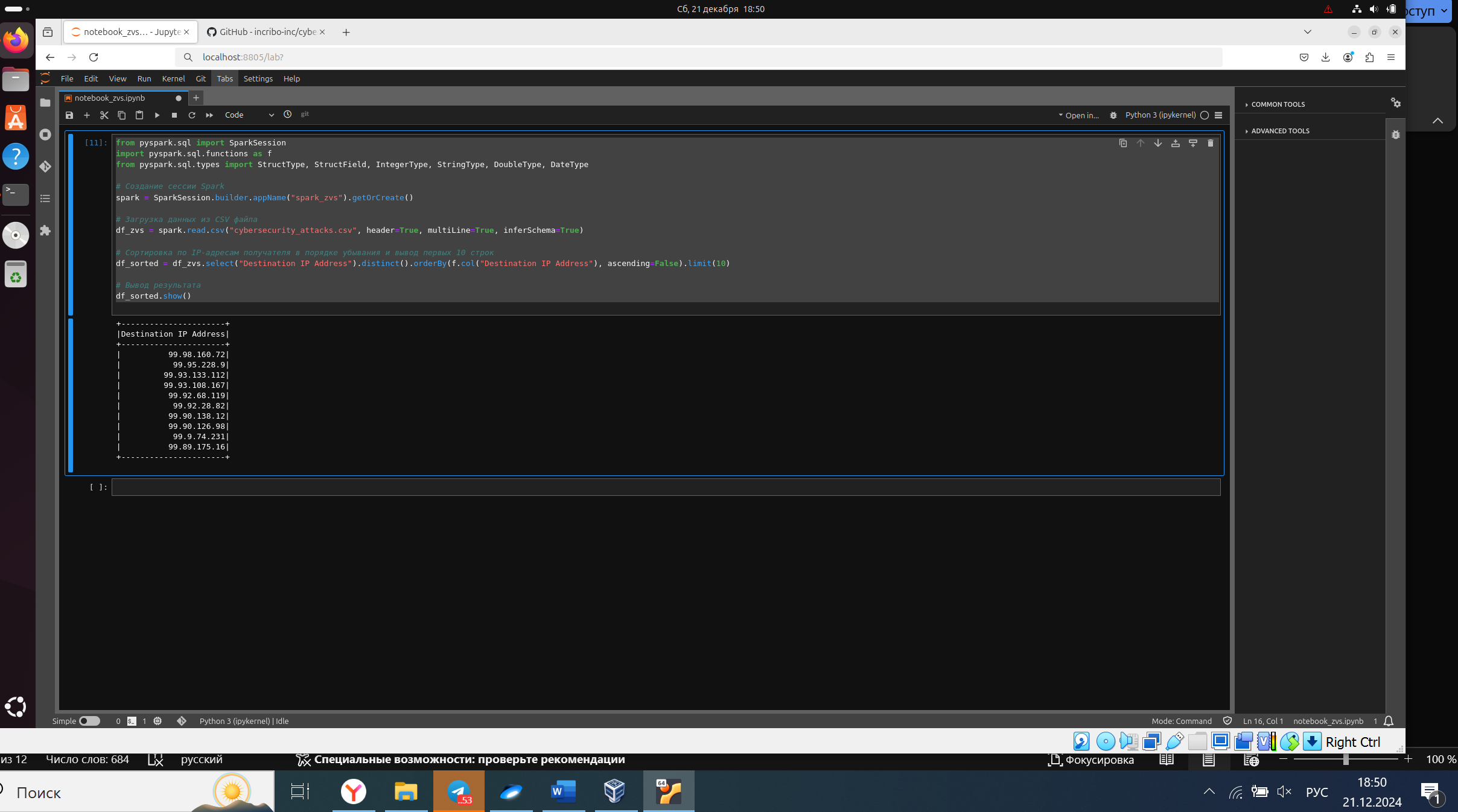
df\_zvs = spark.read.csv("cybersecurity\_attacks.csv", header=True, multiLine=True, inferSchema=True)

# Сортировка по IP-адресам получателя в порядке убывания и вывод первых 10 строк

df\_sorted = df\_zvs.select("Destination IP Address").distinct().orderBy(f.col("Destination IP Address"), ascending=False).limit(10)

# Вывод результата

df\_sorted.show()



Сформируйте свой тестовый набор данных (не менее 7 столбцов и не менее 10000 записей).

# Установка библиотеки faker (если она еще не установлена)

!pip install faker

# Импортируем необходимые библиотеки

import pandas as pd

import random

from faker import Faker

# Создание экземпляра Faker для генерации случайных данных

fake = Faker()

# Количество строк данных

num\_rows = 10000

# Список для хранения сгенерированных данных

data = []

# Генерация данных

for \_ in range(num\_rows):

timestamp = fake.date\_time\_this\_year() # Генерация случайной временной метки

source\_ip = fake.ipv4() # Генерация случайного IP-адреса источника

destination\_ip = fake.ipv4() # Генерация случайного IP-адреса получателя

source\_port = random.randint(1024, 65535) # Генерация случайного порта источника

destination\_port = random.randint(1024, 65535) # Генерация случайного порта получателя

protocol = random.choice(['TCP', 'UDP']) # Выбор случайного протокола

packet\_length = random.randint(40, 1500) # Генерация случайной длины пакета

traffic\_type = random.choice(['Normal', 'Attack']) # Тип трафика (нормальный или атака)

attack\_type = random.choice(['DDoS', 'SQL Injection', 'XSS', 'None']) if traffic\_type == 'Attack' else 'None'

# Добавляем сгенерированные данные в список

data.append([timestamp, source\_ip, destination\_ip, source\_port, destination\_port, protocol, packet\_length, traffic\_type, attack\_type])

# Создаем DataFrame из сгенерированных данных

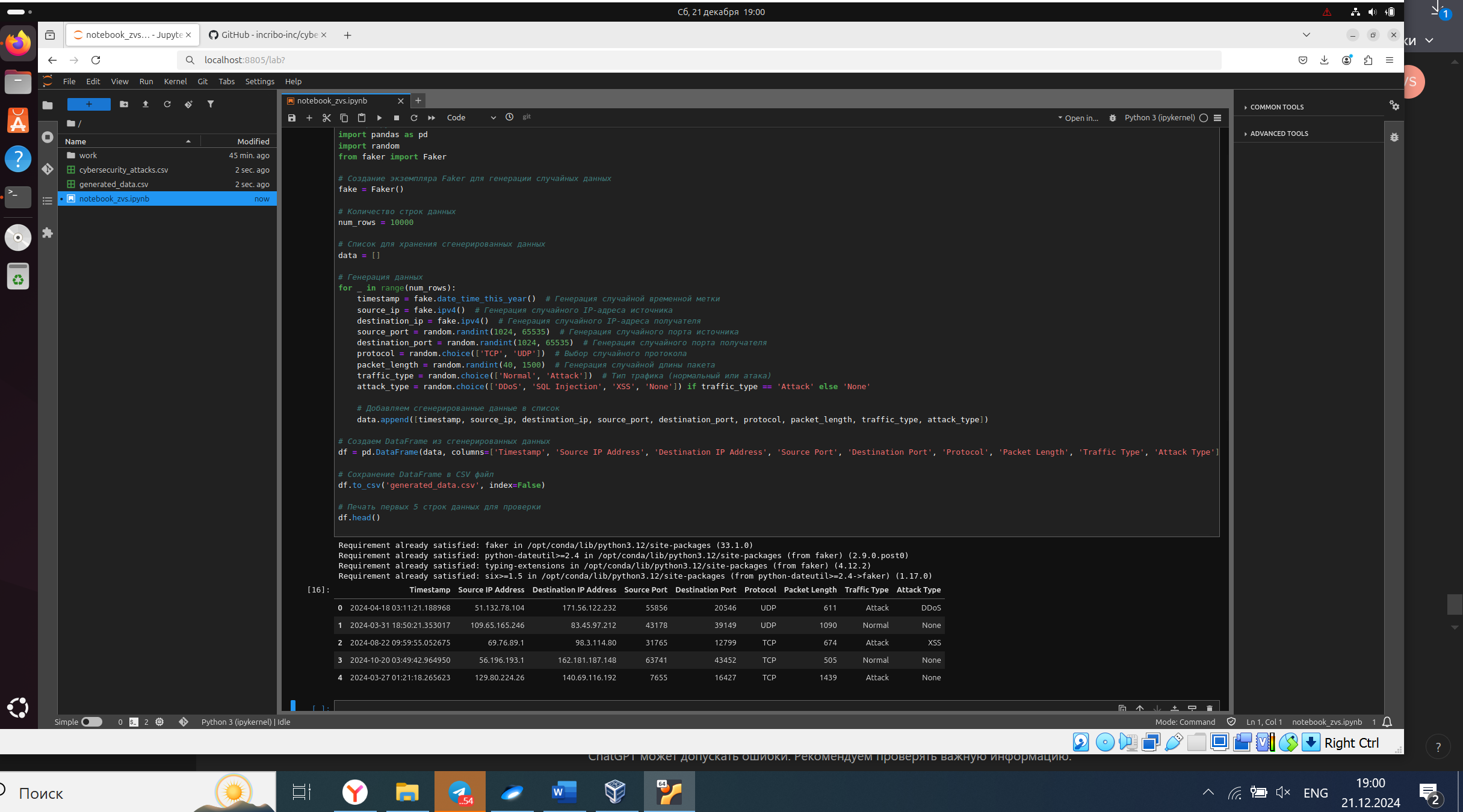
df = pd.DataFrame(data, columns=['Timestamp', 'Source IP Address', 'Destination IP Address', 'Source Port', 'Destination Port', 'Protocol', 'Packet Length', 'Traffic Type', 'Attack Type'])

# Сохранение DataFrame в CSV файл

df.to\_csv('generated\_data.csv', index=False)

# Печать первых 5 строк данных для проверки

df.head()



Проведите не менее 5 манипуляций с новым набором данных с помощью PySpark (с пояснениями и демонстрацией).

Фильтрация данных (атака или нормальная)

from pyspark.sql import SparkSession

import pandas as pd

# Создание Spark сессии

spark = SparkSession.builder.appName("PySpark Data Manipulation").getOrCreate()

# Загрузка данных из CSV в DataFrame

df = pd.read\_csv('generated\_data.csv')

# Конвертация pandas DataFrame в PySpark DataFrame

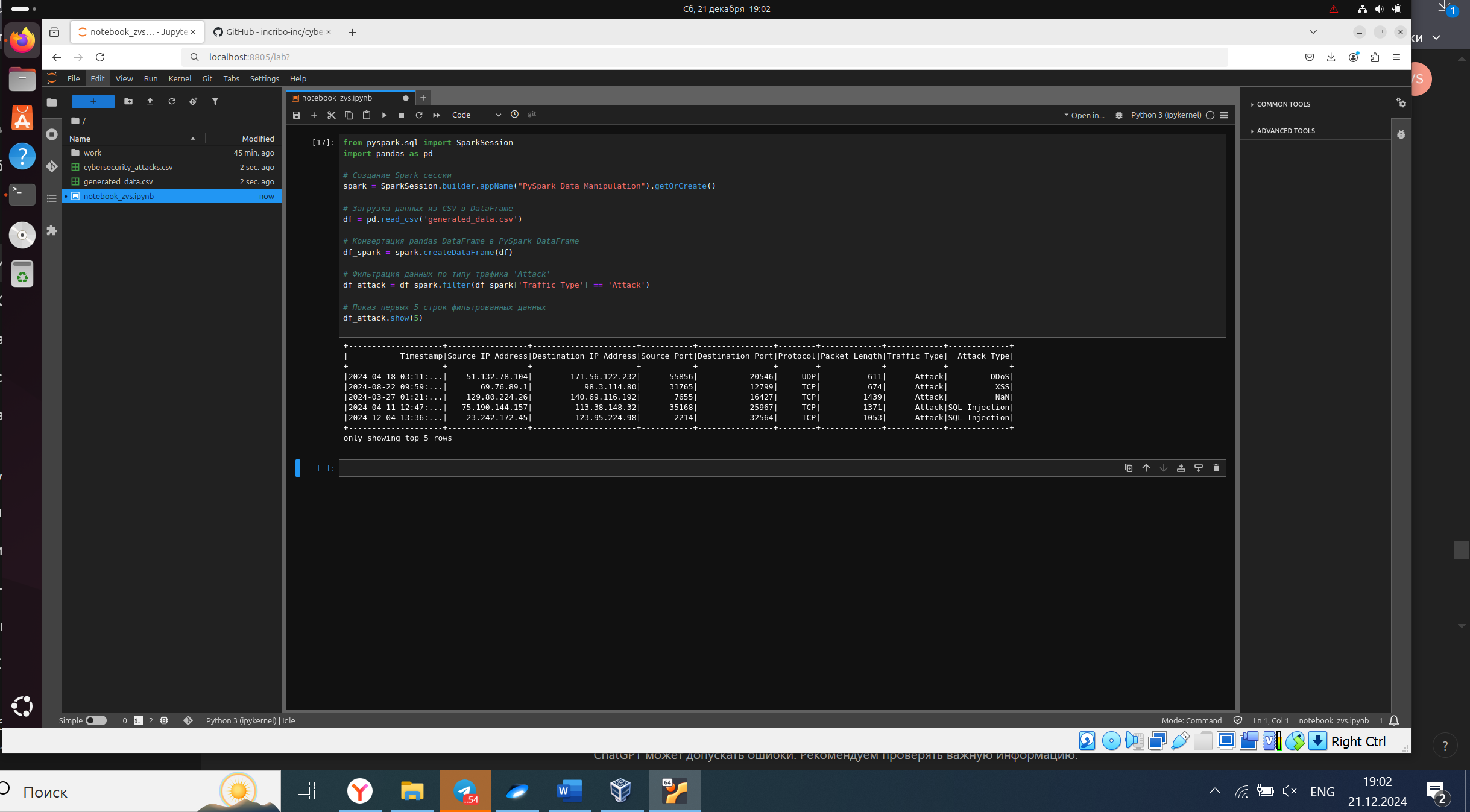
df\_spark = spark.createDataFrame(df)

# Фильтрация данных по типу трафика 'Attack'

df\_attack = df\_spark.filter(df\_spark['Traffic Type'] == 'Attack')

# Показ первых 5 строк фильтрованных данных

df\_attack.show(5)



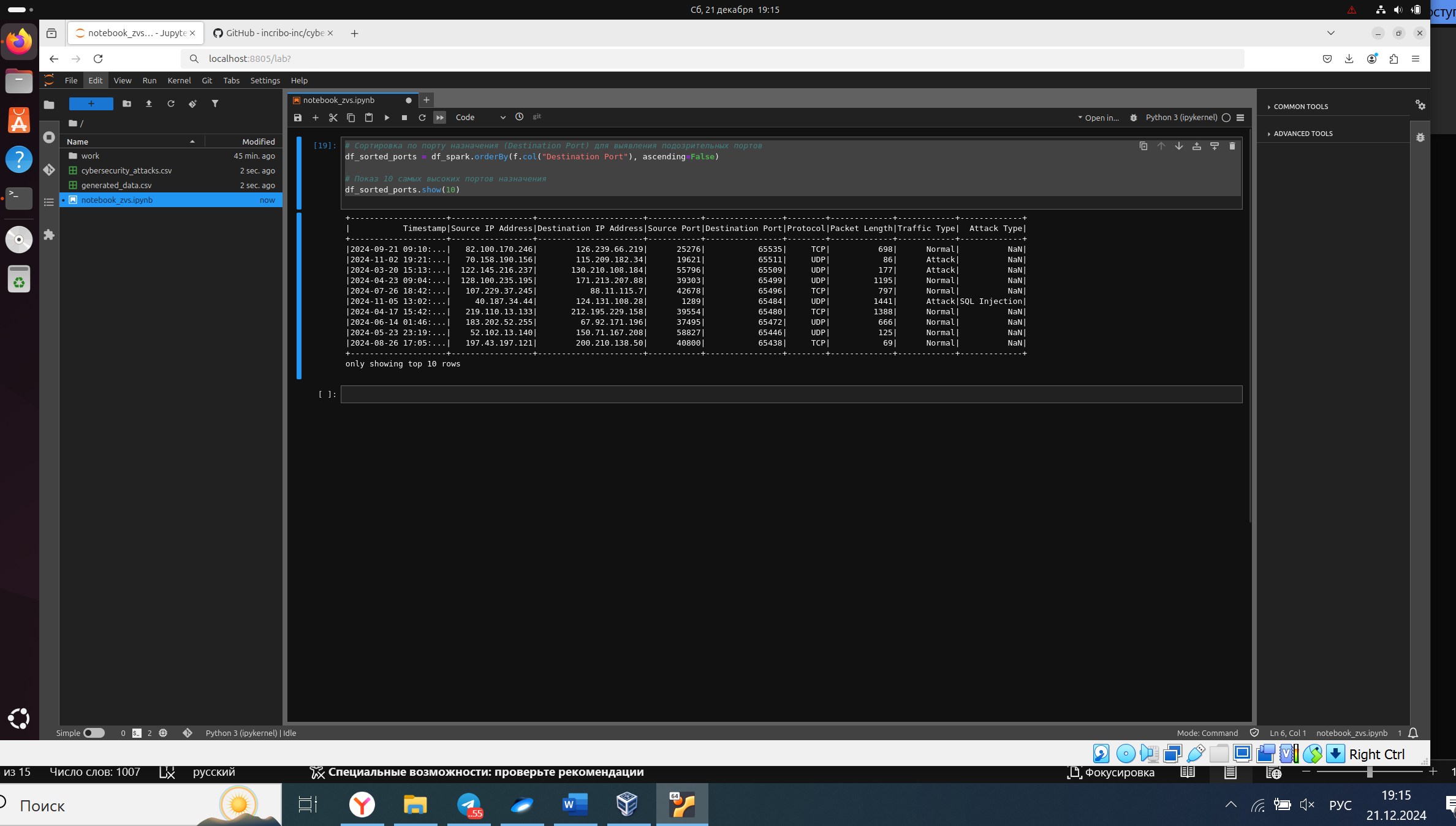
Выявление аномальной активности по портам

# Сортировка по порту назначения (Destination Port) для выявления подозрительных портов

df\_sorted\_ports = df\_spark.orderBy(f.col("Destination Port"), ascending=False)

# Показ 10 самых высоких портов назначения

df\_sorted\_ports.show(10)



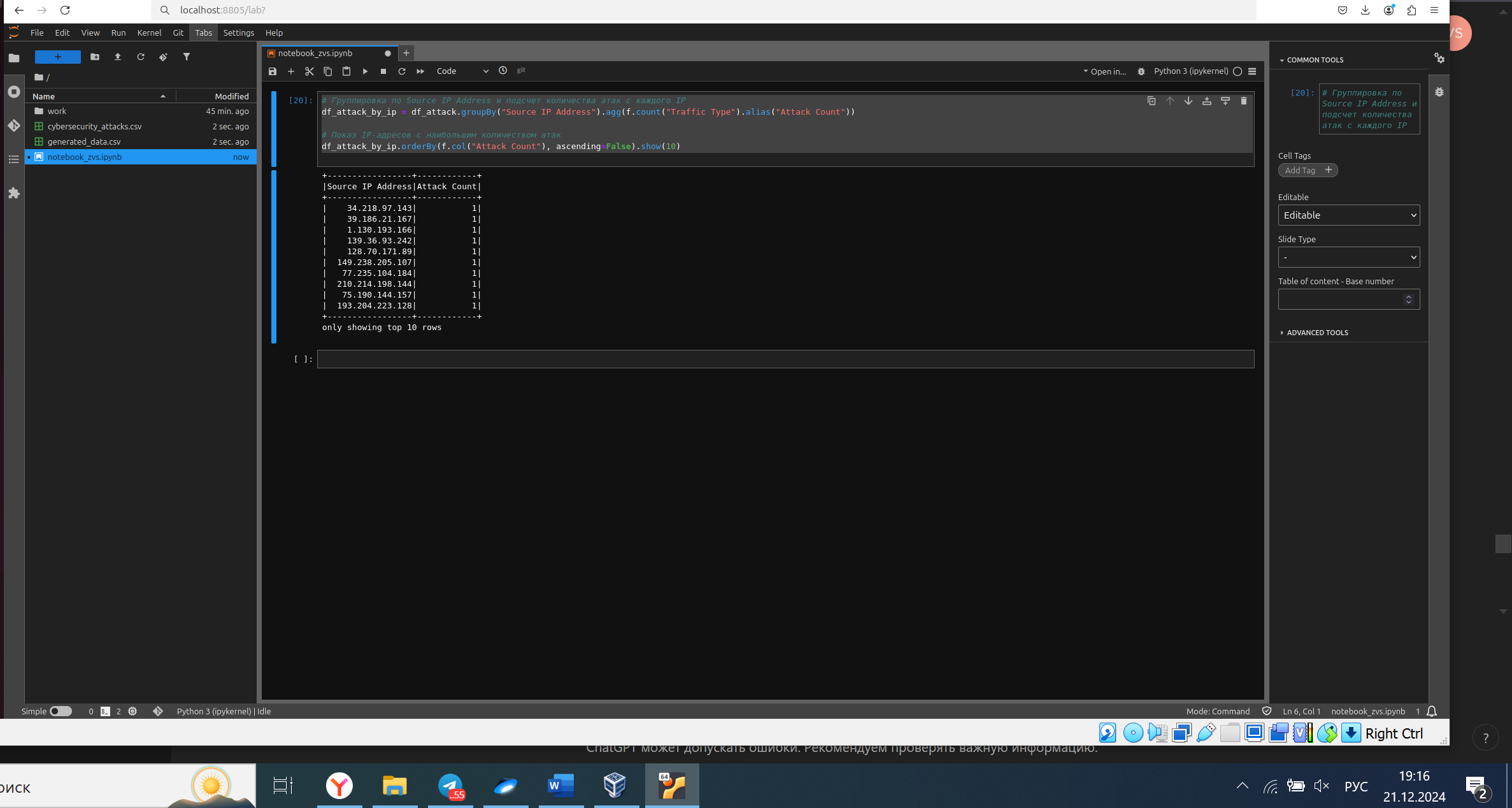
Группировка по IP-адресам для поиска атак по источнику

# Группировка по Source IP Address и подсчет количества атак с каждого IP

df\_attack\_by\_ip = df\_attack.groupBy("Source IP Address").agg(f.count("Traffic Type").alias("Attack Count"))

# Показ IP-адресов с наибольшим количеством атак

df\_attack\_by\_ip.orderBy(f.col("Attack Count"), ascending=False).show(10)



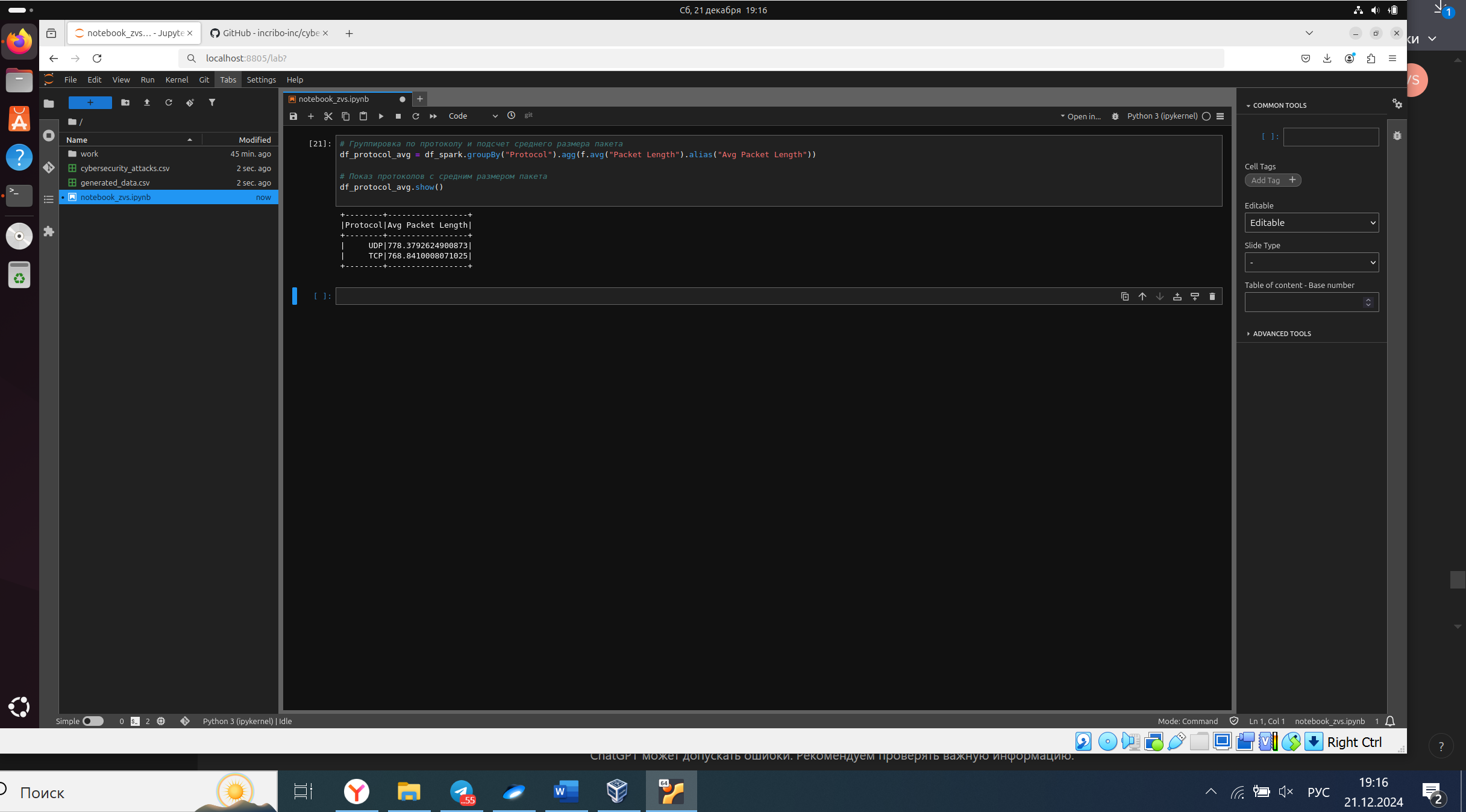
Анализ средних размеров пакетов по протоколу

# Группировка по протоколу и подсчет среднего размера пакета

df\_protocol\_avg = df\_spark.groupBy("Protocol").agg(f.avg("Packet Length").alias("Avg Packet Length"))

# Показ протоколов с средним размером пакета

df\_protocol\_avg.show()



Выявление аномального трафика по временным меткам

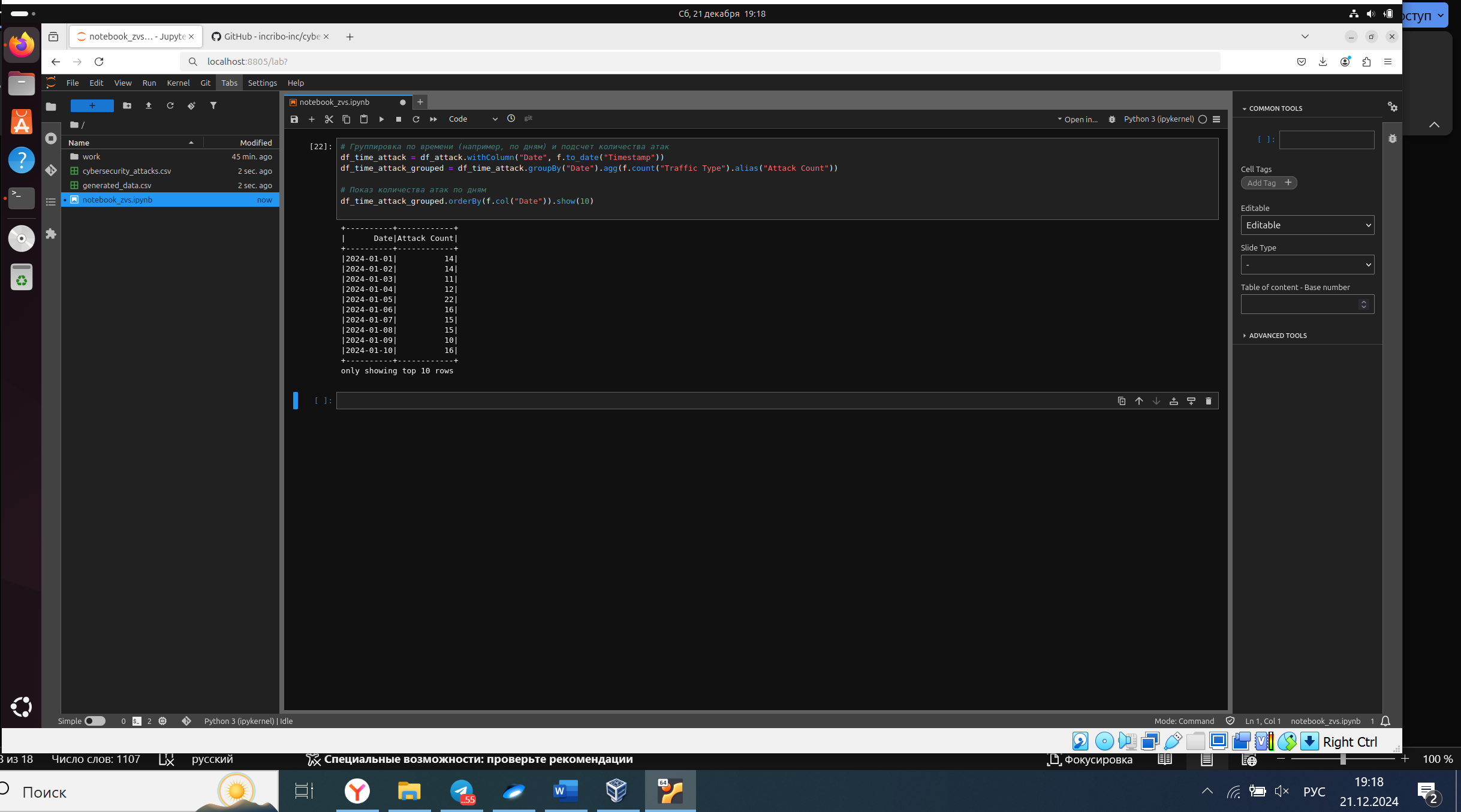
# Группировка по времени (например, по дням) и подсчет количества атак

df\_time\_attack = df\_attack.withColumn("Date", f.to\_date("Timestamp"))

df\_time\_attack\_grouped = df\_time\_attack.groupBy("Date").agg(f.count("Traffic Type").alias("Attack Count"))

# Показ количества атак по дням

df\_time\_attack\_grouped.orderBy(f.col("Date")).show(10)



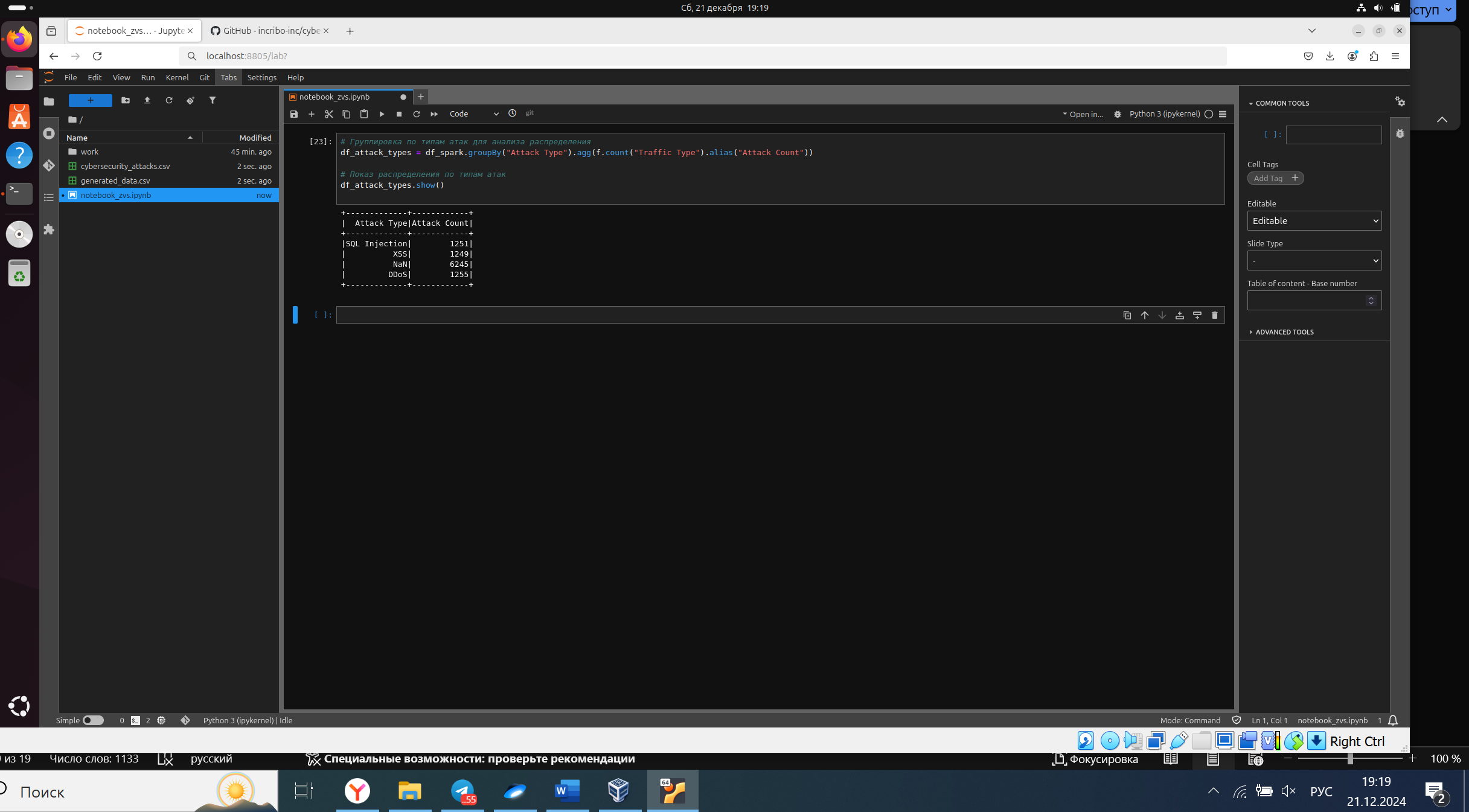
Признаки атаки по типам атак

# Группировка по типам атак для анализа распределения

df\_attack\_types = df\_spark.groupBy("Attack Type").agg(f.count("Traffic Type").alias("Attack Count"))

# Показ распределения по типам атак

df\_attack\_types.show()



Обогащение данных: расчет объемов трафика

# Добавление нового столбца 'Traffic Volume' (объем трафика)

df\_with\_volume = df\_spark.withColumn('Traffic Volume', df\_spark['Packet Length'] \* df\_spark['Destination Port'])

# Показ 5 строк с новыми данными

df\_with\_volume.show(5)

