Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

**Отчёт по лабораторной работе № 5.1**

«Развертывание и настройка кластера Hadoop»

Вариант №8

Выполнила:
студентка группы АДЭУ-211,
Макарова Екатерина Павловна

Преподаватель:
Босенко Тимур Муртазович,
Доцент, кандидат технических
наук

Москва 2024

**Цель**: ознакомление с процессом установки и настройки распределенных систем, таких как Apache(Arenadata) Hadoop. Изучить основные операции и функциональные возможности системы, что позволит понять принципы работы с данными и распределенными вычислениями.

**Необходимое ПО**:

- Ubuntu 24.04 LTS (22.04, 20.04) или новее.
- Java 8 ил Java11 или новее.
- Apache Spark 3.4.3.
- Python 3.12+.
- pip (менеджер пакетов Python).

**Задача**: проанализировать экономические данные, содержащиеся в вашем файле, который находится в файловой системе Hadoop (HDFS). Задача заключается в извлечении, обработке, и анализе данных с целью выявления закономерностей, тенденций, и создания визуализаций на основе предоставленных данных.

**Задание по варианту:**

- Установка Apache(Arenadata) Hadoop и выполнение задачи на объединение файлов в HDFS.
- Данные: Исторические данные по акциям Магнита (MGNT) с сайта Московской биржи (moex.com)
- Операции: Фильтрация данных за 2018 год, расчет средней цены открытия, группировка по годам.

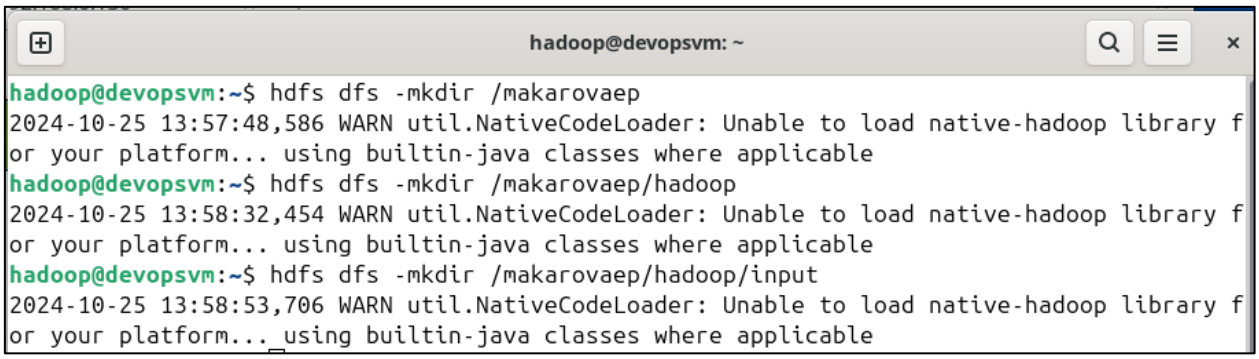
## Практическая часть

1. Запуск Hadoop (Рис. 1).

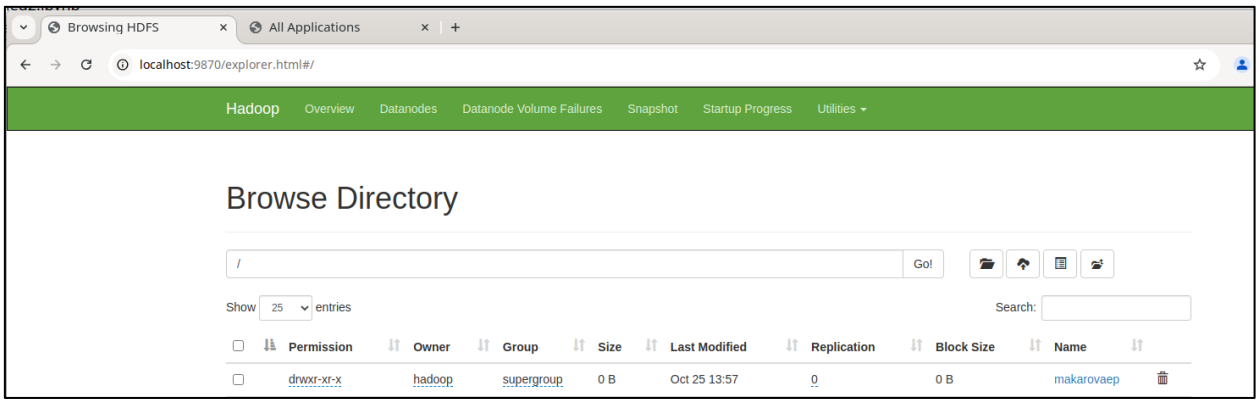
```
hadoop@devopsvm:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [devopsvm]
2024-10-25 13:44:25,983 WARN util.NativeCodeLoader: Unable to load native-hadoop library f
or your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$ ^C
hadoop@devopsvm:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@devopsvm:~$
```

Рис. 1

2. Создание директории в HDFS




3. Проверка создания директории в HDFS




4. Парсинг данных с сайта Московской биржи по акциям Магнита за период с 2006 по 2024 гг.:

На рисунке 2 представлен листинг кода на парсинг данных акций Магнита с сайта московской биржи.

```python
def fetch_data(start_date: str, end_date: str, interval: int) -> pd.DataFrame:

    url = f'http://iss.moex.com/iss/engines/stock/markets/shares/securities/MGNT/candles.json?from={start_date}&till={end_date}&interval={interval}'

    try:
        response = requests.get(url)
        response.raise_for_status()  # Проверка на ошибки HTTP

        # Получаем данные в формате JSON и извлекаем нужные данные
        data = response.json()
        candles = data['candles']['data']
        columns = data['candles']['columns']

        return pd.DataFrame(candles, columns=columns)

    except requests.exceptions.RequestException as e:
        print(f"Ошибка при получении данных: {e}")
        return pd.DataFrame()  # Возвращаем пустой DataFrame в случае ошибки

def collect_data(start_year: int, end_year: int) -> pd.DataFrame:

    all_data = []  # Используем список для временного хранения данных

    # Запрашиваем данные по годам
    for year in range(start_year, end_year + 1):
        start_date = f'{year}-01-01'
        end_date = f'{year}-12-31'

        # Устанавливаем интервал в зависимости от года, тк с 2006 по 2010 данные в интервале суток, после 2010 года данные в интервале секунд
        if year < 2010:
            interval = 24
        else:
            interval = 1

        # Получаем данные за указанный период
        yearly_data = fetch_data(start_date, end_date, interval)

        if not yearly_data.empty:
            all_data.append(yearly_data)  # Добавляем DataFrame в список

    return pd.concat(all_data, ignore_index=True) if all_data else pd.DataFrame()  # Объединяем все DataFrame

if __name__ == "__main__":
    start_year = 2006  # Начальный год
    end_year = 2024  # Конечный год

    all_data = collect_data(start_year, end_year)

    # Проверяем, были ли получены данные, и выводим их
    if not all_data.empty:
        print("Данные успешно загружены:")
        print(all_data)
    else:
```

Рис. 2

Вывод результата парсинга представлен на рисунке 3.

```
[11]: all_data
```

[11]:

| | open | close | high | low | value | volume | begin | end |
|---|---|---|---|---|---|---|---|---|
| **0** | 810.00 | 730.0 | 810.0 | 730.00 | 478000.0 | 600 | 2006-05-16 00:00:00 | 2006-05-16 23:59:59 |
| **1** | 638.00 | 636.0 | 641.0 | 633.00 | 1273800.0 | 2000 | 2006-06-07 00:00:00 | 2006-06-07 23:59:59 |
| **2** | 635.26 | 644.0 | 644.0 | 630.13 | 573178.0 | 900 | 2006-06-08 00:00:00 | 2006-06-08 23:59:59 |
| **3** | 600.00 | 597.0 | 600.0 | 595.00 | 1849000.0 | 3100 | 2006-06-14 00:00:00 | 2006-06-14 23:59:59 |
| **4** | 595.00 | 597.0 | 620.0 | 595.00 | 3905000.0 | 6500 | 2006-06-15 00:00:00 | 2006-06-15 23:59:59 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **7835** | 7155.00 | 7155.0 | 7155.0 | 7154.50 | 1144798.5 | 160 | 2024-01-03 18:15:00 | 2024-01-03 18:15:59 |
| **7836** | 7155.00 | 7155.0 | 7155.0 | 7154.50 | 1438149.5 | 201 | 2024-01-03 18:16:00 | 2024-01-03 18:16:59 |
| **7837** | 7154.50 | 7154.5 | 7155.0 | 7154.00 | 9178646.0 | 1283 | 2024-01-03 18:17:00 | 2024-01-03 18:17:59 |
| **7838** | 7154.50 | 7150.0 | 7154.5 | 7149.50 | 2889454.5 | 404 | 2024-01-03 18:18:00 | 2024-01-03 18:18:59 |
| **7839** | 7151.50 | 7149.0 | 7151.5 | 7149.00 | 1008037.0 | 141 | 2024-01-03 18:19:00 | 2024-01-03 18:19:59 |

7840 rows × 8 columns

Рис. 3

Просмотрим информацию о полученных данных (Рис. 4).

```
[12]: all_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7840 entries, 0 to 7839
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   open    7840 non-null   float64
 1   close   7840 non-null   float64
 2   high    7840 non-null   float64
 3   low     7840 non-null   float64
 4   value   7840 non-null   float64
 5   volume  7840 non-null   int64
 6   begin   7840 non-null   object
 7   end     7840 non-null   object
dtypes: float64(5), int64(1), object(2)
memory usage: 490.1+ KB
```

Рис. 4

Объём полученных данных (490.1 КБ) не подходит под условие задачи, поэтому необходимо сгенерировать данные в размере минимум 128МБ.

5. Обработка данных:

Видим, что атрибуты “begin” и “end”, которые обозначают дату и время открытия и закрытия торгов, имеют тип «object». Данные столбцы необходимо изменить в тип datetime (Рис. 5).

```
all_data['begin']=pd.to_datetime(all_data['begin'])
all_data['end']=pd.to_datetime(all_data['end'])

all_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7840 entries, 0 to 7839
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   open    7840 non-null   float64
 1   close   7840 non-null   float64
 2   high    7840 non-null   float64
 3   low     7840 non-null   float64
 4   value   7840 non-null   float64
 5   volume  7840 non-null   int64
 6   begin   7840 non-null   datetime64[ns]
 7   end     7840 non-null   datetime64[ns]
dtypes: datetime64[ns](2), float64(5), int64(1)
memory usage: 490.1 KB
```

Рис. 5

Статистические показатели цифровых столбцов (Рис. 6).

```
all_data.describe()
```

| | open | close | high | low | value | volume | begin | end |
|---|---|---|---|---|---|---|---|---|
| count | 7840.000000 | 7840.000000 | 7840.000000 | 7840.000000 | 7.840000e+03 | 7840.000000 | 7840 | 7840 |
| mean | 5747.855246 | 5748.256967 | 5752.527324 | 5742.858337 | 2.896749e+06 | 1504.123342 | 2016-08-01 16:27:29.992346880 | 2016-08-01 19:02:35.115306240 |
| min | 290.000000 | 290.500000 | 300.000000 | 260.040000 | 2.639600e+03 | 0.000000 | 2006-05-16 00:00:00 | 2006-05-16 23:59:59 |
| 25% | 3441.000000 | 3441.000000 | 3442.000000 | 3440.500000 | 2.033112e+05 | 36.000000 | 2013-01-08 12:35:45 | 2013-01-08 12:36:44 |
| 50% | 5467.250000 | 5467.750000 | 5468.750000 | 5465.750000 | 7.404827e+05 | 129.000000 | 2017-01-03 11:20:30 | 2017-01-03 11:21:29 |
| 75% | 8986.175000 | 8982.225000 | 8989.900000 | 8980.025000 | 2.286804e+06 | 424.000000 | 2021-01-04 10:38:15 | 2021-01-04 10:39:14 |
| max | 11474.000000 | 11473.000000 | 11476.000000 | 11468.000000 | 1.358322e+08 | 113962.000000 | 2024-01-03 18:19:00 | 2024-01-03 18:19:59 |
| std | 3182.824168 | 3183.093471 | 3181.906539 | 3183.817838 | 7.719216e+06 | 6141.775964 | NaN | NaN |

Рис. 6

Проверка на нулевые значения (Рис. 7).

```
null_values = all_data.isnull()
print(null_values)
```

```
       open  close   high    low  value  volume  begin    end
0     False  False  False  False  False   False  False  False
1     False  False  False  False  False   False  False  False
2     False  False  False  False  False   False  False  False
3     False  False  False  False  False   False  False  False
4     False  False  False  False  False   False  False  False
...     ...    ...    ...    ...    ...     ...    ...    ...
7835  False  False  False  False  False   False  False  False
7836  False  False  False  False  False   False  False  False
7837  False  False  False  False  False   False  False  False
7838  False  False  False  False  False   False  False  False
7839  False  False  False  False  False   False  False  False

[7840 rows x 8 columns]
```

Рис. 7

После того, как данные были обработаны, сохраним датафрейм в csv файл (Рис. 8).

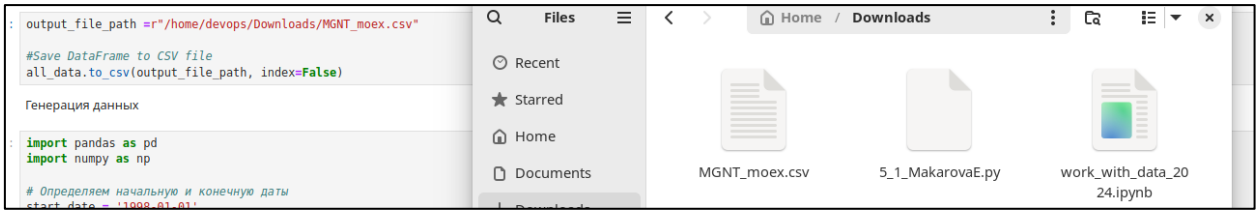



Рис. 8

Проверка размера файла (Рис. 9).


Рис. 9

Так как объём данных, полученных с сайта, не подходит под условие, сгенерируем данные для задачи.

6. Генерация данных

На рисунке представлен листинг кода на генерацию данных (Рис. 10).

Генерация данных

```
import pandas as pd
import numpy as np

# Определяем начальную и конечную даты
start_date = '1998-01-01'
end_date = '2005-12-31'

# Генерируем диапазон дат с интервалом в 1 минуту
date_range = pd.date_range(start=start_date, end=end_date, freq='min')

# Определяем количество строк
num_rows = len(date_range)

# Генерируем случайные данные для open, close, high, low, value и volume
np.random.seed(42)  # Для воспроизводимости
open_prices = np.random.uniform(7000, 8000, size=num_rows).astype(np.float64)
close_prices = np.random.uniform(7000, 8000, size=num_rows).astype(np.float64)
high_prices = np.maximum(open_prices, close_prices) + np.random.uniform(0, 100, size=num_rows).astype(np.float64)
low_prices = np.minimum(open_prices, close_prices) - np.random.uniform(0, 100, size=num_rows).astype(np.float64)
values = np.random.uniform(1e5, 5e5, size=num_rows).astype(np.float64)
volumes = np.random.randint(100, 1000, size=num_rows).astype(np.int64)

# Создаем DataFrame
df = pd.DataFrame({
    'open': open_prices,
    'close': close_prices,
    'high': high_prices,
    'low': low_prices,
    'value': values,
    'volume': volumes,
    'begin': date_range,
    'end': date_range + pd.Timedelta(minutes=1)  # Устанавливаем end на 1 минуту позже begin
})

# Проверяем результат
print(df.info())
print(df.head())
```

Рис. 10

Результат генерации данных (Рис. 11).

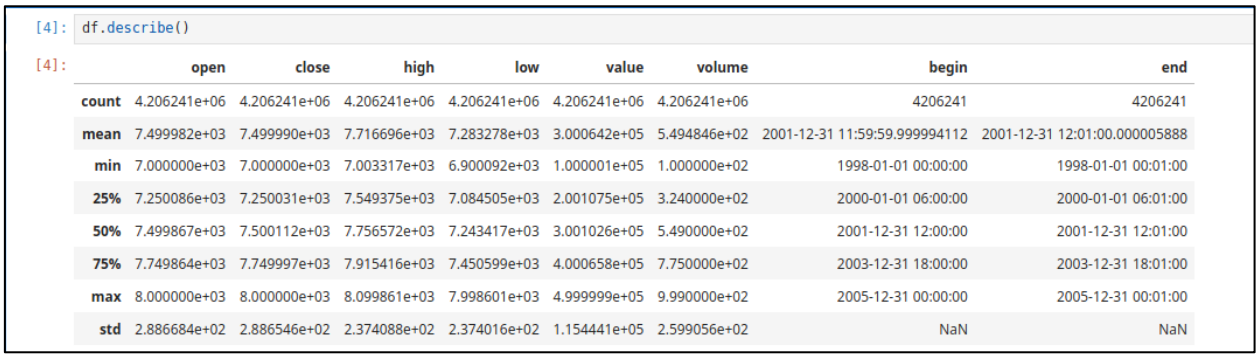
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4206241 entries, 0 to 4206240
Data columns (total 8 columns):
 #   Column  Dtype
---  ------  -----
 0   open    float64
 1   close   float64
 2   high    float64
 3   low     float64
 4   value   float64
 5   volume  int64
 6   begin   datetime64[ns]
 7   end     datetime64[ns]
dtypes: datetime64[ns](2), float64(5), int64(1)
memory usage: 256.7 MB
None
          open        close         high          low          value  volume  \
0  7374.540119  7191.195798  7435.293270  7110.239729  341280.955384     779
1  7950.714306  7723.742202  7950.891037  7659.136750  203780.705195     333
2  7731.993942  7123.955488  7809.942463  7087.586172  127621.825303     156
3  7598.658484  7600.047565  7696.805112  7581.824063  217868.418140     688
4  7156.018640  7858.388755  7945.396863  7111.926342  332321.455499     492

                begin                 end
0 1998-01-01 00:00:00 1998-01-01 00:01:00
1 1998-01-01 00:01:00 1998-01-01 00:02:00
2 1998-01-01 00:02:00 1998-01-01 00:03:00
3 1998-01-01 00:03:00 1998-01-01 00:04:00
4 1998-01-01 00:04:00 1998-01-01 00:05:00
```

Рис. 11

Объём данных подходит для задачи – 256.7 МБ.

Статистические показатели по числовым столбцам (Рис. 12)



[4]:
```
df.describe()
```

[4]:

| | open | close | high | low | value | volume | begin | end |
|---|---|---|---|---|---|---|---|---|
| count | 4.206241e+06 | 4.206241e+06 | 4.206241e+06 | 4.206241e+06 | 4.206241e+06 | 4.206241e+06 | 4206241 | 4206241 |
| mean | 7.499982e+03 | 7.499990e+03 | 7.716696e+03 | 7.283278e+03 | 3.000642e+05 | 5.494846e+02 | 2001-12-31 11:59:59.999994112 | 2001-12-31 12:01:00.000005888 |
| min | 7.000000e+03 | 7.000000e+03 | 7.003317e+03 | 6.900092e+03 | 1.000001e+05 | 1.000000e+02 | 1998-01-01 00:00:00 | 1998-01-01 00:01:00 |
| 25% | 7.250086e+03 | 7.250031e+03 | 7.549375e+03 | 7.084505e+03 | 2.001075e+05 | 3.240000e+02 | 2000-01-01 06:00:00 | 2000-01-01 06:01:00 |
| 50% | 7.499867e+03 | 7.500112e+03 | 7.756572e+03 | 7.243417e+03 | 3.001026e+05 | 5.490000e+02 | 2001-12-31 12:00:00 | 2001-12-31 12:01:00 |
| 75% | 7.749864e+03 | 7.749997e+03 | 7.915416e+03 | 7.450599e+03 | 4.000658e+05 | 7.750000e+02 | 2003-12-31 18:00:00 | 2003-12-31 18:01:00 |
| max | 8.000000e+03 | 8.000000e+03 | 8.099861e+03 | 7.998601e+03 | 4.999999e+05 | 9.990000e+02 | 2005-12-31 00:00:00 | 2005-12-31 00:01:00 |
| std | 2.886684e+02 | 2.886546e+02 | 2.374088e+02 | 2.374016e+02 | 1.154441e+05 | 2.599056e+02 | NaN | NaN |

Рис. 12

Сохранение датафрейма в csv файла (Рис. 13).

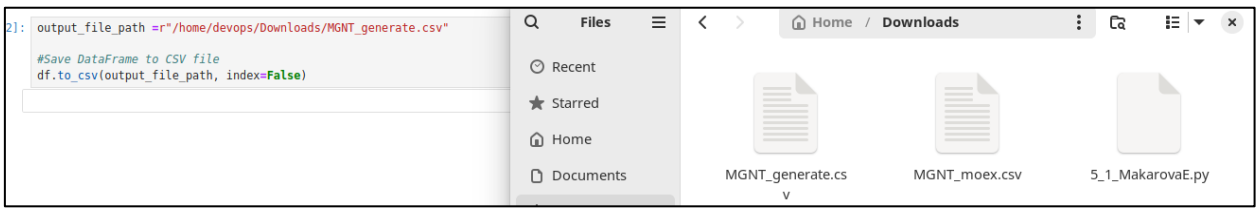



Рис. 13

Проверка размера файла (Рис. 14).

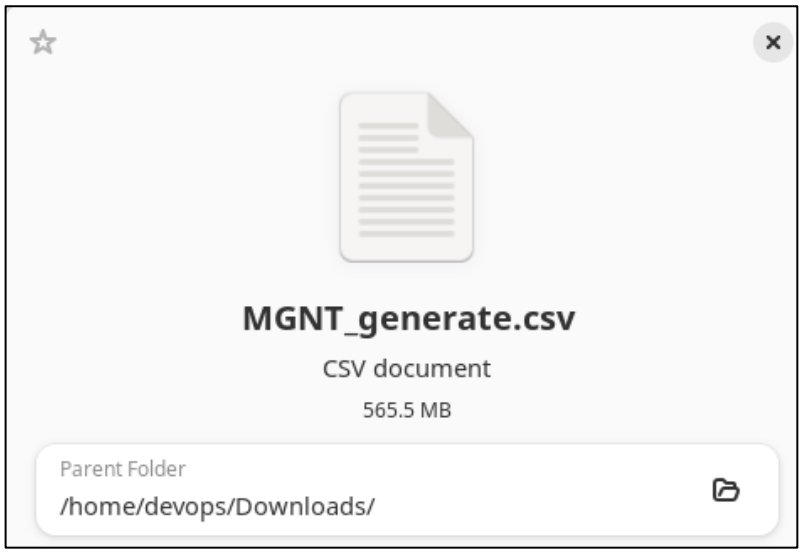



Рис. 14

Данный размер подходит под условие задачи, загрузим оба файла в HDFS. Для этого необходимо загрузить файлы в github, чтобы в дальнейшем перенести их на Hadoop

7. Загрузка данных в HDFS

1) Перенос файла с использованием временной директории

Выполнение команд на копирование файла во временную директорию и предоставление доступа представлено на рисунке 16.

```
devops@devopsvm:~$ cp /home/devops/MGNT_generate.csv /tmp/
devops@devopsvm:~$ chmod 777 /tmp/MGNT_generate.csv
devops@devopsvm:~$
```

Рис. 15

Копирование файла из временной директории в домашнюю директорию пользователя Hadoop представлено на рисунке 17.

```
hadoop@devopsvm:/home/devops$ sudo cp /tmp/MGNT_generate.csv /home/hadoop/
[sudo] password for hadoop:
hadoop@devopsvm:/home/devops$
```

Рис. 16

Загрузка файла MGNT_generate.csv в директорию HDFS (Рис. 17).

```
hadoop@devopsvm:~$ hadoop fs -put MGNT_generate.csv /makarovaep/hadoop/input/
2024-11-17 00:20:06,359 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform..
. using builtin-java classes where applicable
hadoop@devopsvm:~$
```

Рис. 17

Получение файла MGNT_moex.csv из репозитория в GitHub (Рис. 18).

```
hadoop@devopsvm:~$ wget https://raw.githubusercontent.com/MakarovaKatya22/Big_DATA/refs/heads/main/MGNT_moex
.csv
--2024-11-17 00:21:10--  https://raw.githubusercontent.com/MakarovaKatya22/Big_DATA/refs/heads/main/MGNT_moe
x.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199
.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 639735 (625K) [text/plain]
Saving to: 'MGNT_moex.csv'

MGNT_moex.csv             100%[=======================================>] 624.74K   807KB/s    in 0.8s

2024-11-17 00:21:11 (807 KB/s) - 'MGNT_moex.csv' saved [639735/639735]

hadoop@devopsvm:~$
```

Рис. 18

Проверка двух загруженных файлов в директории пользователя Hadoop (Рис. 19).

```
hadoop@devopsvm:~$ ls
Desktop    GDP.csv              MGNT_generate.csv  output    snap                          thinclient_drives
Documents  hadoop-3.3.5.tar.gz  MGNT_moex.csv      Pictures  spark-3.4.3-bin-hadoop3.tgz   Videos
Downloads  hdfs                 Music              Public    Templates
hadoop@devopsvm:~$
```

Рис. 19

Загрузка файла MGNT_moex.csv в директорию HDFS (Рис. 20).

```
hadoop@devopsvm:~$ hadoop fs -put MGNT_moex.csv /makarovaep/hadoop/input/
2024-11-17 00:24:20,059 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your pla
tform... using builtin-java classes where applicable
hadoop@devopsvm:~$
```

Рис. 20

Проверка загруженных файлов в директории HDFS (Рис. 21).

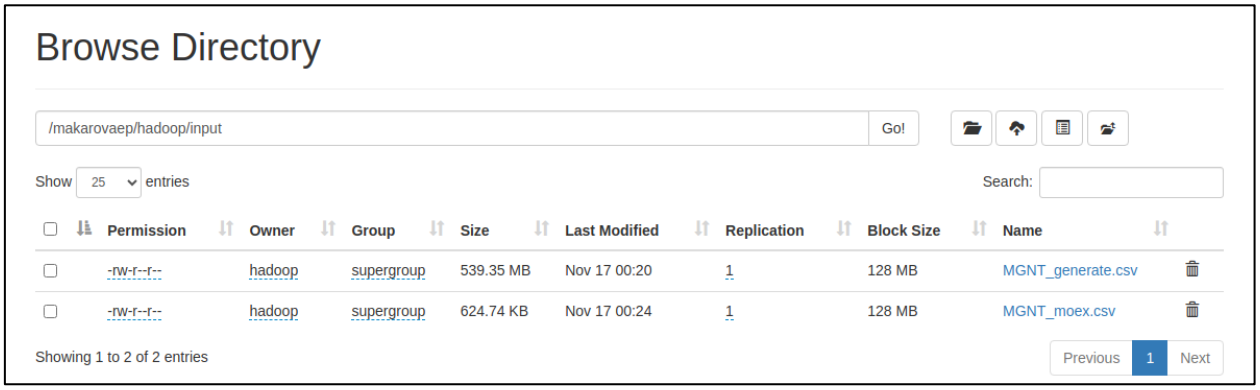



Рис. 21

Установка прав на директорию в hdfs (Рис. 22).

```
hadoop@devopsvm:~$ hdfs dfs -chmod 777 /makarovaep/hadoop/input
2024-11-17 00:37:14,444 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
 applicable
hadoop@devopsvm:~$
```

Рис. 22

8. Получение данных с помощью pyspark из hdfs:

Установка pyspark и установка необходимых библиотек, а также создание сессии в spark (Рис. 23).

```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in ./.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (3.5.3)
Requirement already satisfied: py4j==0.10.9.7 in ./.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from
pyspark) (0.10.9.7)
```

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
from pyspark.sql import SparkSession

# Создание SparkSession
spark = SparkSession.builder \
    .appName("MGNT Data Analysis") \
    .config("spark.hadoop.fs.defaultFS", "hdfs://localhost:9000") \
    .config("spark.ui.port", "4050") \
    .getOrCreate()

# Установка количества разделов для shuffle операций
spark.conf.set("spark.sql.shuffle.partitions", "50")
```

Рис. 23

Получение первого файла из hdfs и вывод первых 5-ти строк (Рис. 24).

```
# Чтение данных из HDFS
file_path_1= "hdfs://localhost:9000/makarovaep/hadoop/input/MGNT_moex.csv"
df1 = spark.read.csv(file_path_1, header=True, inferSchema=True)

# Просмотр первых строк данных
df1.show(5)
```

```
+------+-----+-----+------+---------+------+-------------------+-------------------+
|  open|close| high|   low|    value|volume|              begin|                end|
+------+-----+-----+------+---------+------+-------------------+-------------------+
| 810.0|730.0|810.0| 730.0| 478000.0|   600|2006-05-16 00:00:00|2006-05-16 23:59:59|
| 638.0|636.0|641.0| 633.0|1273800.0|  2000|2006-06-07 00:00:00|2006-06-07 23:59:59|
|635.26|644.0|644.0|630.13| 573178.0|   900|2006-06-08 00:00:00|2006-06-08 23:59:59|
| 600.0|597.0|600.0| 595.0|1849000.0|  3100|2006-06-14 00:00:00|2006-06-14 23:59:59|
| 595.0|597.0|620.0| 595.0|3905000.0|  6500|2006-06-15 00:00:00|2006-06-15 23:59:59|
+------+-----+-----+------+---------+------+-------------------+-------------------+
only showing top 5 rows
```

Рис. 24

Получение второго файла из hdfs и вывод первых 5-ти строк (Рис. 25).

```
# Чтение данных из HDFS
file_path2 = "hdfs://localhost:9000/makarovaep/hadoop/input/MGNT_generate.csv"
df2 = spark.read.csv(file_path2, header=True, inferSchema=True)

# Просмотр первых строк данных
df2.show(5)
```

```
+------------------+------------------+------------------+------------------+------------------+------+-------------------+-------------------+
|              open|             close|              high|               low|             value|volume|              begin|                end|
+------------------+------------------+------------------+------------------+------------------+------+-------------------+-------------------+
|7374.5401188473625|7191.195797689377|7435.293269611488|7110.239729328441|  341280.955384198|   779|1998-01-01 00:00:00|1998-01-01 00:01:00|
| 7950.714306409916|7723.742202310935|7950.891037090337|7659.136749568901|203780.70519476736|   333|1998-01-01 00:01:00|1998-01-01 00:02:00|
| 7731.993941811405| 7123.95548837091|7809.942462632805|7087.586171959409|127621.82530305526|   156|1998-01-01 00:02:00|1998-01-01 00:03:00|
| 7598.658484197036|7600.047564796336|7696.805112466492| 7581.82406295645|217868.41813971289|   688|1998-01-01 00:03:00|1998-01-01 00:04:00|
| 7156.018640442437|7858.388754621724|7945.396862639947|7111.926342019782|332321.45549878245|   492|1998-01-01 00:04:00|1998-01-01 00:05:00|
+------------------+------------------+------------------+------------------+------------------+------+-------------------+-------------------+
only showing top 5 rows
```

Рис. 25

9. Обработка данных:

Просмотр структуры данных двух датасетов (Рис. 26).

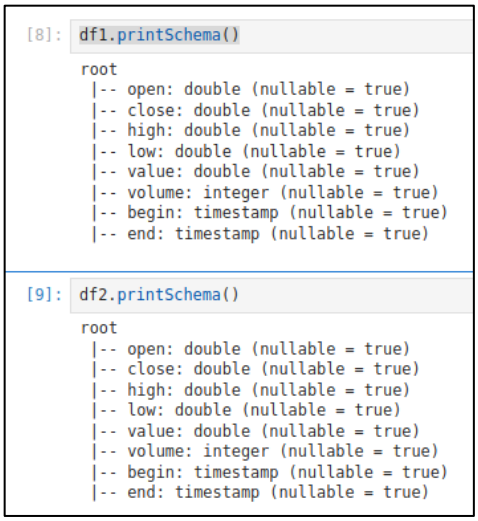



Рис. 26

Данные имеют одинаковый тип, поэтому их можно объединить.

Описание данных:

| Наименование атрибута | Описание |
|---|---|
| Open | Это цена открытия акции на начало торгового интервала. Она указывает на первую цену, по которой была совершена сделка в этот интервал. |
| Close | Это цена закрытия акции в конце торгового дня. Она представляет собой последнюю цену, по которой была совершена сделка перед закрытием рынка. |
| High | Это максимальная цена, достигнутая акцией в течение торгового интервала. Этот показатель помогает определить пиковую активность и интерес к акции. |
| Low | Это минимальная цена, зафиксированная акцией в течение торгового интервала. Этот показатель показывает наименьшую цену, по которой акция была продана. |
| Value | Этот столбец обычно указывает на общую стоимость всех сделок, совершенных с акцией в течение интервала. Это может быть рассчитано как произведение цены и объема. |
| Volume | Это количество акций, которые были куплены и проданы в течение торгового дня. Этот показатель помогает оценить ликвидность акции. |
| Begin | Этот столбец может указывать на время начала торгового дня или период, за который представлены данные (например, дата начала). |
| End | Этот столбец может указывать на время закрытия торгового дня или период, за который представлены данные (например, дата окончания). |

Объединение датасетов и просмотр статистической информации по числовым столбцам (Рис. 27).

```
r_df = df1.union(df2)
r_df.describe().show()
```

```
[Stage 9:================================================>        (5 + 1) / 6]
+-------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|summary|             open|            close|             high|              low|            value|           volume|
+-------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|  count|          4214081|          4214081|          4214081|          4214081|          4214081|          4214081|
|   mean|7496.722455038049| 7496.731358963318| 7713.041891300093| 7280.411682498173|304895.1653804528| 551.2606262195719|
| stddev|328.2067510994055|328.19553505154437|286.80265137683523|281.98802935864467|369683.1866552452|373.21130891109016|
|    min|            290.0|            290.5|            300.0|            260.04|           2639.6|                0|
|    max|          11474.0|          11473.0|          11476.0|          11468.0|   1.3583221825E8|           113962|
+-------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```

Рис. 27

Проверка на пропуски (Рис. 28).

```
# Подсчет пропусков
missing_values = r_df.select([sum(col(c).isNull().cast("int")).alias(c) for c in r_df.columns])
missing_values.show()
```

```
[Stage 12:================================================>        (5 + 1) / 6]
+----+-----+----+---+-----+------+-----+---+
|open|close|high|low|value|volume|begin|end|
+----+-----+----+---+-----+------+-----+---+
|   0|    0|   0|  0|    0|     0|    0|  0|
+----+-----+----+---+-----+------+-----+---+
```

Рис. 28

10. Анализ результатов статистических показателей с рисунка 27:

1) Количество записей (count): Все столбцы содержат 4,214,081 записей, что указывает на полный набор данных без пропусков.

2) Цены акций

- Средняя цена открытия (mean open): 7496.72
- Средняя цена закрытия (mean close): 7496.73

Цены открытия и закрытия очень близки друг к другу, что может свидетельствовать о стабильности акций в течение анализируемого периода.

- Максимальная цена (max high): 7713.04
- Минимальная цена (min low): 260.04

Разница между максимальной и минимальной ценой значительна (максимум — 11474.0, минимум — 290.0), что указывает на высокую изменчивость акций в течение рассматриваемого периода.

3) Статистические показатели

Стандартное отклонение:

- Для цен открытия и закрытия оно составляет около 328.21, что говорит о том, что цены колебались относительно среднего значения.
- Для максимальной и минимальной цен стандартное отклонение ниже (286.80 и 281.99 соответственно), что также подтверждает изменчивость.

4) Объем торгов

- Средний объем (mean volume): 551.26
- Максимальный объем (max volume): 113962

Средний объем торгов относительно низкий по сравнению с максимальным значением, что может указывать на наличие периодов высокой активности на рынке.

5) Общая стоимость сделок

- Средняя стоимость сделок (mean value): 304895.17
- Максимальная стоимость сделок (max value): 135832218.25

Высокая средняя стоимость сделок может свидетельствовать о значительном интересе инвесторов к акциям Магнита.

**Вывод**: Данные показывают, что акции MGNT демонстрируют высокую волатильность с достаточно стабильными ценами открытия и закрытия. Объем торгов варьируется, с периодами высокой активности, что может быть связано с новостями или событиями на рынке.

11. Агрегирование данных:

1) Фильтрация данных за 2018 год (Рис. 29):

```
#фильтрация данных за 2018 год
filtered_df = r_df.filter((col("begin") >= "2018-01-01") & (col("begin") < "2019-01-01"))
filtered_df.show()
```

```
+------+------+------+------+-----------+------+-------------------+-------------------+
|  open| close|  high|   low|      value|volume|              begin|                end|
+------+------+------+------+-----------+------+-------------------+-------------------+
|6327.0|6327.0|6327.0|6327.0|   145521.0|    23|2018-01-03 09:59:00|2018-01-03 09:59:59|
|6327.0|6342.0|6358.0|6322.0|  4131181.0|   652|2018-01-03 10:00:00|2018-01-03 10:00:59|
|6343.0|6368.0|6368.0|6343.0|  2384402.0|   375|2018-01-03 10:01:00|2018-01-03 10:01:59|
|6368.0|6392.0|6399.0|6368.0|1.1909342E7|  1865|2018-01-03 10:02:00|2018-01-03 10:02:59|
|6392.0|6391.0|6392.0|6385.0|  1654945.0|   259|2018-01-03 10:03:00|2018-01-03 10:03:59|
|6392.0|6432.0|6434.0|6392.0|1.0093677E7|  1575|2018-01-03 10:04:00|2018-01-03 10:04:59|
|6433.0|6420.0|6450.0|6407.0|  7793155.0|  1212|2018-01-03 10:05:00|2018-01-03 10:05:59|
|6410.0|6408.0|6425.0|6406.0|  4388582.0|   684|2018-01-03 10:06:00|2018-01-03 10:06:59|
|6417.0|6407.0|6419.0|6407.0|  2800668.0|   437|2018-01-03 10:07:00|2018-01-03 10:07:59|
|6409.0|6405.0|6410.0|6404.0|  3254517.0|   508|2018-01-03 10:08:00|2018-01-03 10:08:59|
|6405.0|6404.0|6405.0|6404.0|  1562751.0|   244|2018-01-03 10:09:00|2018-01-03 10:09:59|
|6402.0|6400.0|6403.0|6399.0|  1657806.0|   259|2018-01-03 10:10:00|2018-01-03 10:10:59|
|6397.0|6399.0|6402.0|6397.0|  1529657.0|   239|2018-01-03 10:11:00|2018-01-03 10:11:59|
|6398.0|6402.0|6402.0|6397.0|  1401403.0|   219|2018-01-03 10:12:00|2018-01-03 10:12:59|
|6402.0|6405.0|6406.0|6402.0|  1664847.0|   260|2018-01-03 10:13:00|2018-01-03 10:13:59|
|6406.0|6399.0|6410.0|6399.0|  1249126.0|   195|2018-01-03 10:14:00|2018-01-03 10:14:59|
|6403.0|6403.0|6405.0|6403.0|   365021.0|    57|2018-01-03 10:15:00|2018-01-03 10:15:59|
|6403.0|6400.0|6404.0|6400.0|   544305.0|    85|2018-01-03 10:16:00|2018-01-03 10:16:59|
|6400.0|6399.0|6400.0|6399.0|   230378.0|    36|2018-01-03 10:17:00|2018-01-03 10:17:59|
|6400.0|6393.0|6403.0|6393.0|  1873367.0|   293|2018-01-03 10:18:00|2018-01-03 10:18:59|
+------+------+------+------+-----------+------+-------------------+-------------------+
only showing top 20 rows
```

Рис. 29

Просмотр статистических показателей за 2018 год (Рис. 30).

```
filtered_df.describe().show()
```

```
[Stage 16:================================================>        (5 + 1) / 6]
+-------+------------------+------------------+------------------+------------------+------------------+------------------+
|summary|              open|             close|              high|               low|             value|            volume|
+-------+------------------+------------------+------------------+------------------+------------------+------------------+
|  count|               500|               500|               500|               500|               500|               500|
|   mean|          6460.316|          6460.856|          6463.094|          6458.29|      1686699.846|           261.138|
| stddev|24.118780548574936|23.337620982145612|22.702775072488 23|24.48708096667447|2275086.990601439|352.0180853604566|
|    min|            6327.0|            6327.0|            6327.0|            6322.0|            6400.0|                 1|
|    max|            6484.0|            6483.0|            6485.0|            6482.0|         2.3212319E7|              3581|
+-------+------------------+------------------+------------------+------------------+------------------+------------------+
```

Рис. 30

- Средняя цена открытия (mean open): 6460.32
- Средняя цена закрытия (mean close): 6460.86

Цены открытия и закрытия очень близки друг к другу, что может свидетельствовать о стабильности акций в рассматриваемом периоде.

- Максимальная цена (max high): 6485.00
- Минимальная цена (min low): 6322.00

Разница между максимальной и минимальной ценой составляет 163.00, что указывает на умеренную волатильность акций в течение анализируемого периода.

2) Расчет средней цены открытия (Рис. 31):

```
from pyspark.sql.functions import avg
# Расчет средней цены открытия
average_open_price = r_df.agg(avg("open").alias("average_open")).collect()[0]["average_open"]

# Вывод результата
print(f"Средняя цена открытия: {average_open_price}")
```

```
[Stage 19:==================================================>        (5 + 1) / 6]
Средняя цена открытия: 7496.722455038049
```

Рис. 31

Полученное значение соответствует данным, полученным с помощью функции describe (Рис. 27).

3) Группировка по годам и вычисление средних значений по числовым столбцам (Рис. 32):

```
from pyspark.sql.functions import year, avg, round
# Группировка данных по годам и расчет среднего значения для всех числовых столбцов
average_values_by_year = r_df.groupBy(year("begin").alias("year")) \
    .agg(
        round(avg("open"), 2).alias("average_open"),
        round(avg("close"), 2).alias("average_close"),
        round(avg("high"), 2).alias("average_high"),
        round(avg("low"), 2).alias("average_low"),
        round(avg("value"), 2).alias("average_value"),
        round(avg("volume"), 2).alias("average_volume")
    ) \
    .orderBy("year")

# Вывод результата
average_values_by_year.show()
```

```
[Stage 28:==================================================>        (5 + 1) / 6]
+----+------------+-------------+------------+-----------+-------------+--------------+
|year|average_open|average_close|average_high|average_low|average_value|average_volume|
+----+------------+-------------+------------+-----------+-------------+--------------+
|1998|     7500.13|      7499.89|     7716.74|    7283.36|    300139.53|        549.61|
|1999|     7500.39|       7500.2|     7717.05|    7283.53|    299877.36|        549.81|
|2000|     7499.49|      7499.99|     7716.54|    7282.91|    300109.35|        549.28|
|2001|     7499.74|      7499.45|     7716.12|    7283.01|    300156.29|        549.08|
|2002|     7499.91|      7500.29|     7716.92|    7283.34|    299947.36|         549.8|
|2003|     7500.53|      7499.67|     7716.71|    7283.48|    300184.43|        549.74|
|2004|     7499.71|      7499.99|     7716.46|    7283.16|    300275.86|        549.65|
|2005|     7499.96|      7500.45|     7717.03|    7283.45|    299822.17|         548.9|
|2006|      811.53|       816.39|       822.9|     806.52|   1470361.23|       1733.33|
|2007|     1118.32|      1119.65|     1136.31|    1102.84|    8911350.6|       7827.02|
|2008|       944.2|       942.17|      961.17|     921.67|1.394773882E7|      14030.41|
|2009|     1281.26|       1283.6|     1304.51|    1253.49| 2.14606008E7|      17381.48|
|2011|      2717.5|      2717.44|     2720.98|    2714.12|    531663.22|        195.64|
|2012|     2861.92|       2862.1|     2863.26|    2860.44|    234607.82|         82.14|
|2013|     5005.33|      5005.32|     5008.08|    5003.08|   1062653.71|        212.14|
|2014|     9070.39|      9069.63|     9074.67|    9064.58|   1840225.08|        203.41|
|2015|     9962.57|       9967.1|     9971.82|     9955.2|   2262037.08|        225.83|
|2016|    11192.77|     11193.05|    11197.55|   11187.84|   2228602.39|        198.46|
|2017|    11319.11|      11319.4|     11325.6|   11310.37|   2731369.23|        241.49|
|2018|     6460.32|      6460.86|     6463.09|    6458.29|   1686699.85|        261.14|
+----+------------+-------------+------------+-----------+-------------+--------------+
only showing top 20 rows
```

Рис. 32

12. Визуализация данных

Визуализация временных рядов (Рис. 33):

Группировка по годам и вычисление средних значений были выполнены на предыдущем этапе, поэтому мы можем визуализировать полученные данные.

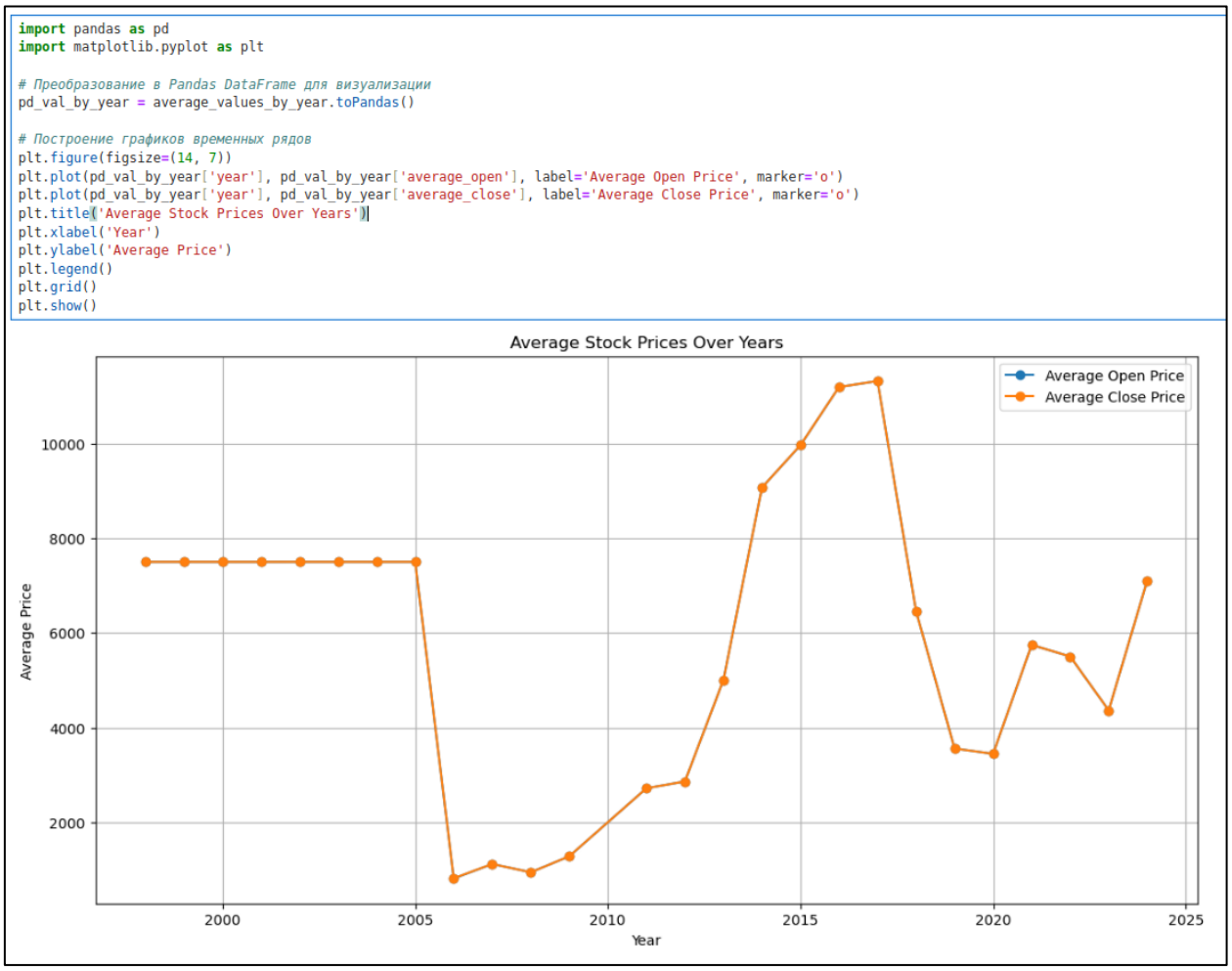



Рис. 33

Из полученной визуализации можно сделать вывод, что средняя цена открытия и закрытия примерно одинаковая.

С 1998 года по 2005 год средние цены открытия и закрытия достаточно стабильны, это связано с тем, что данные за указанный период были сгенерированы. С 2006 года наблюдается рост цен открытия и закрытия до 2017 года, после этого наблюдается спад, что может быть связано с негативными событиями на рынке.

После 2020 года наблюдается нестабильное состояние цен открытия и закрытия.

Листинг кода для визуализации изменения значения среднего объёма по годам (Рис. 34).

```
# Построение горизонтальной столбчатой диаграммы для среднего объема
plt.figure(figsize=(14, 7))
bars = plt.barh(pd_val_by_year['year'], pd_val_by_year['average_volume'], color='skyblue')

# Добавление меток и заголовка
plt.title('Average Trading Volume by Year')
plt.xlabel('Average Volume')
plt.ylabel('Year')
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Показать значения рядом со столбцами
for bar in bars:
    width = bar.get_width()  # Получаем ширину столбца (средний объем)
    plt.text(width + 10, bar.get_y() + bar.get_height()/2, f'{width:.2f}',
             va='center')  # Добавляем текст справа от столбца

# Показать график
plt.show()
```

Рис. 34

Результат выполнения кода (Рис. 35).

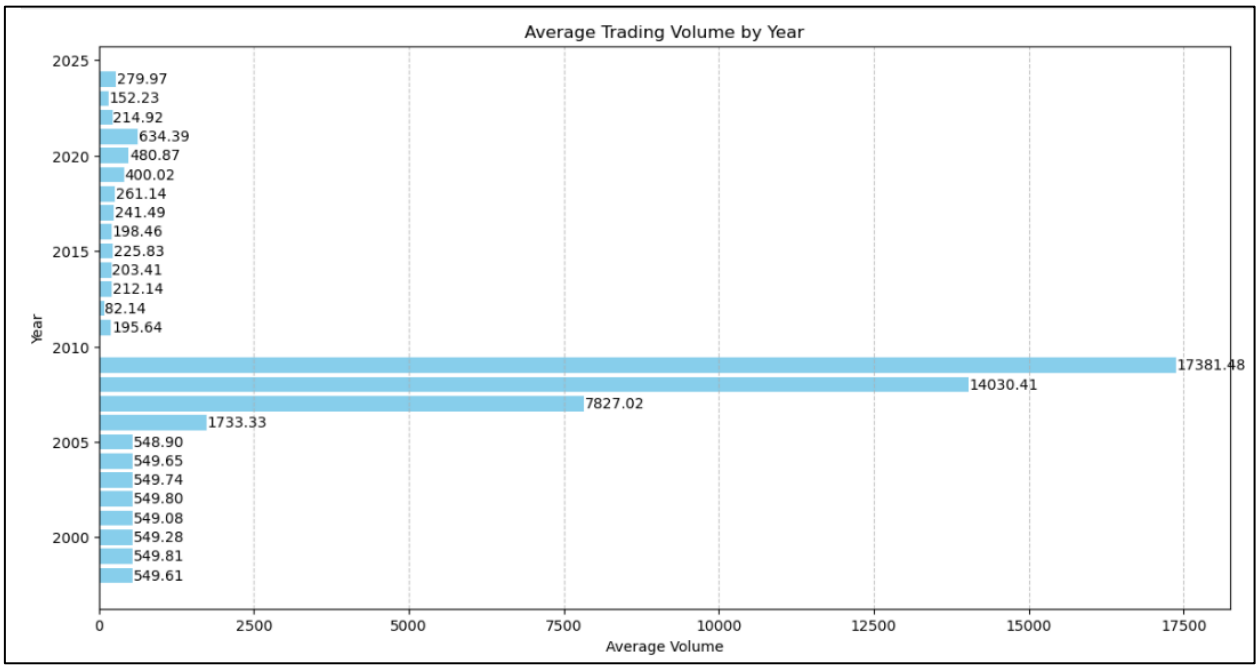



Рис. 35

Аналогично средним ценам открытия/закрытия в период с 1998 по 2005гг. среднее значение объема – стабильно, в 2006 году наблюдается рост, который заканчивается в 2010 году, что указывает на высокий интерес инвесторов.

В период с 2011 по 2025 среднее значение объёма имеет небольшое колебание.

13. Загрузка полученных датасетов в hdfs:

1) Загрузка объединённого датасета из двух (Рис. 36)

```
# Путь в HDFS для сохранения
file_path_hdfs1 = "hdfs://localhost:9000/makarovaep/hadoop/input/MGNT_union_file.csv"

# Сохранение DataFrame в формате CSV в HDFS
r_df.write.csv(file_path_hdfs1, header=True, mode='overwrite')
```

Рис. 36

Проверка загруженных данных в директории hdfs (Рис. 37).

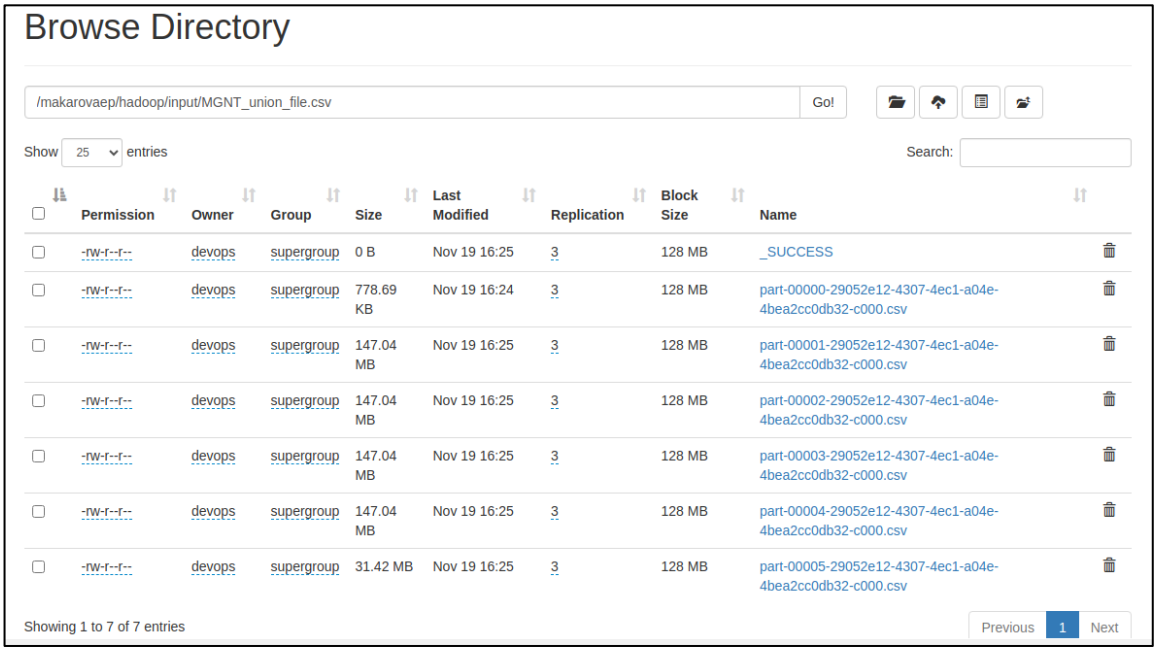



Рис. 37

2) Загрузка датасета с данными за 2018 год (Рис. 38)

```
# Путь в HDFS для сохранения
file_path_hdfs2 = "hdfs://localhost:9000/makarovaep/hadoop/input/MGNT_2018.csv"

# Сохранение DataFrame в формате CSV в HDFS
filtered_df.write.csv(file_path_hdfs2, header=True, mode='overwrite')
```

Рис. 38

Проверка директории загруженного файла с данными за 2018 год (Рис. 39).

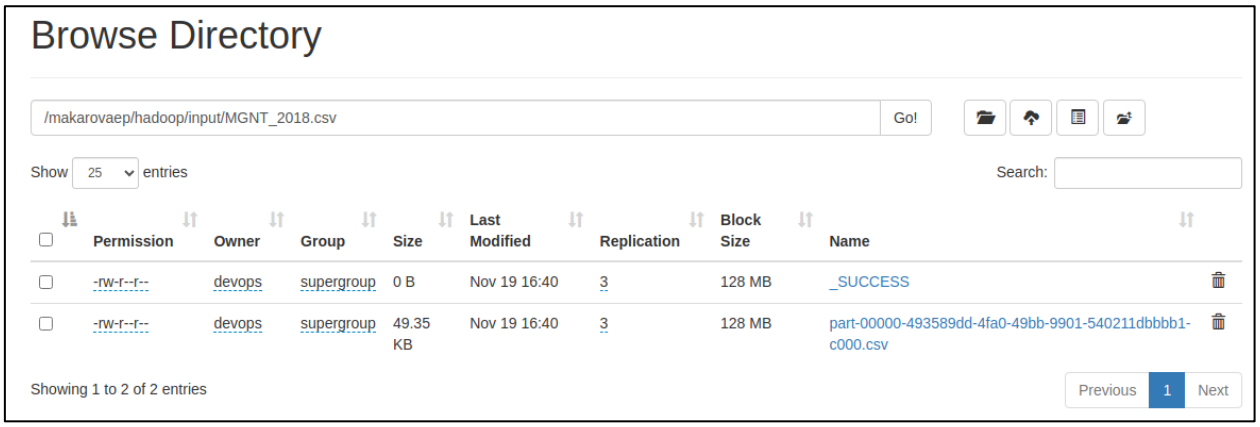



Рис. 39

3) Загрузка датасета с данными, сгруппированными по годам, и средними значениями (Рис. 40).

```
# Путь в HDFS для сохранения
file_path_hdfs3 = "hdfs://localhost:9000/makarovaep/hadoop/input/MGNT_average_values_by_year.csv"

# Сохранение DataFrame в формате CSV в HDFS
average_values_by_year.write.csv(file_path_hdfs3, header=True, mode='overwrite')
```

Рис. 40

Проверка директории загруженного файла в hdfs (Рис. 41).

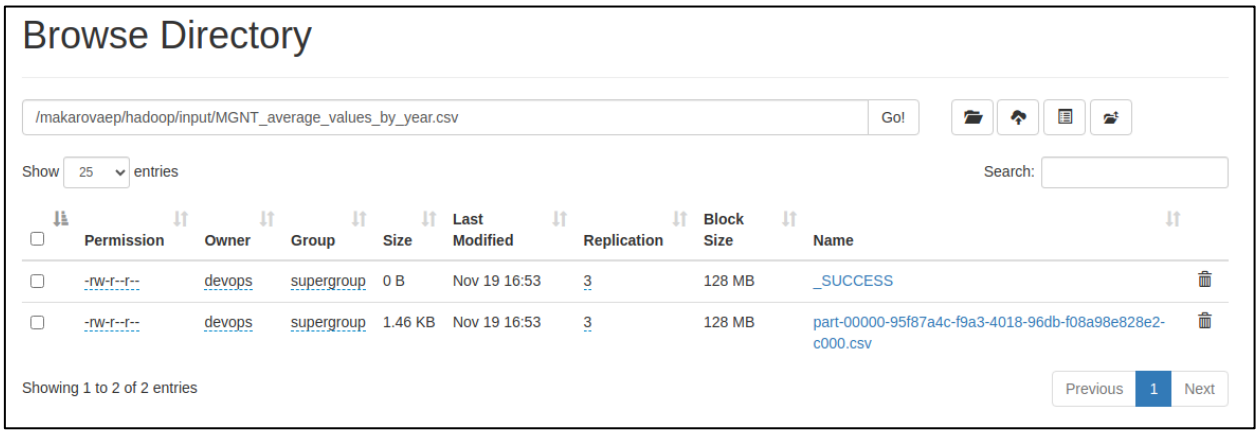



Рис. 41