

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

Ли Александр Андреевич БД-241м

**Практическая работа 1. Введение в большие данные и их хранение.
Инструменты обработки больших данных (Hadoop)**

Направление подготовки/специальность

38.04.05 - Бизнес-информатика

Бизнес-аналитика и большие данные

(очная форма обучения)

Вариант 12

Москва

2024

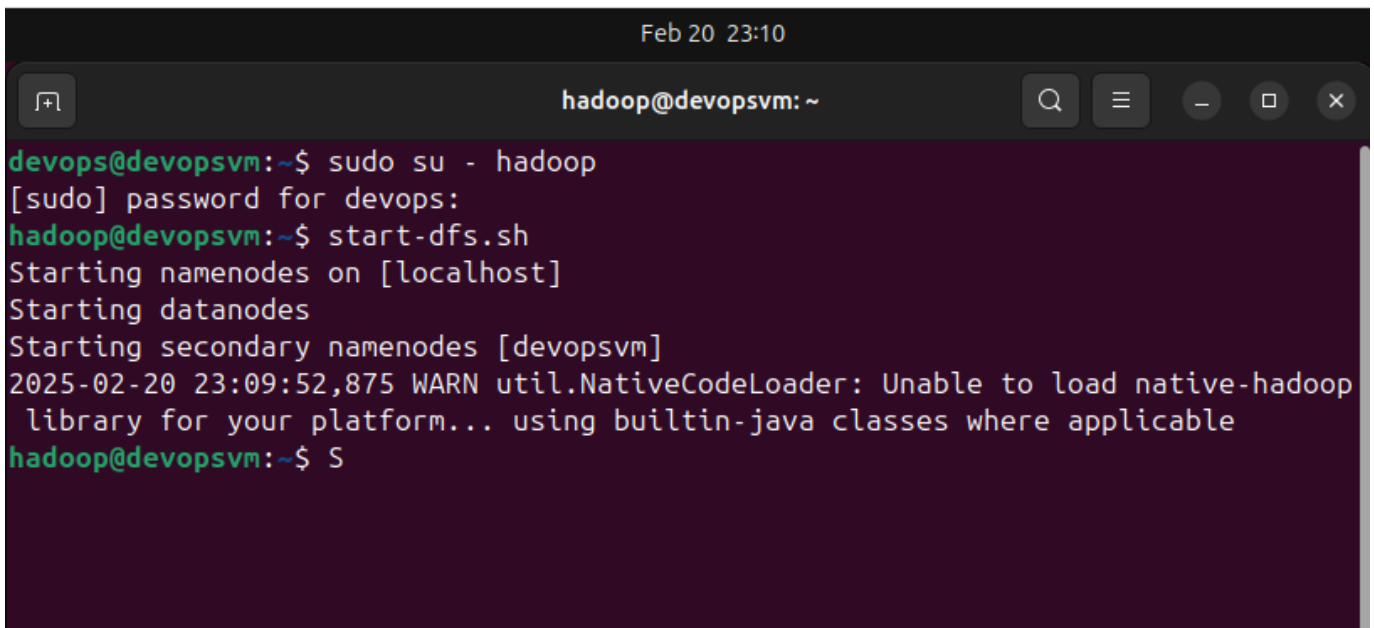
Цель

Изучить основные операции и функциональные возможности системы, что позволит понять принципы работы с данными и распределенными вычислениями.

Основная часть

Запускаем hadoop

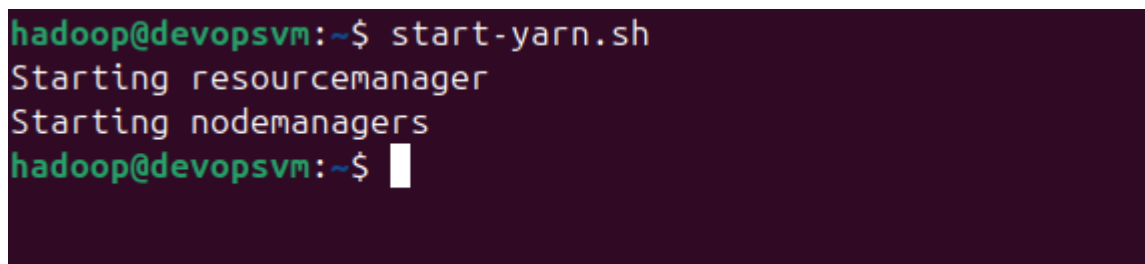
Start-dfs.sh

A terminal window titled 'Feb 20 23:10' with the prompt 'hadoop@devopsvm: ~'. The user enters 'sudo su - hadoop', then '[sudo] password for devops:'. The prompt changes to 'hadoop@devopsvm:~\$'. The user enters 'start-dfs.sh'. The output shows 'Starting namenodes on [localhost]', 'Starting datanodes', and 'Starting secondary namenodes [devopsvm]'. A warning message follows: '2025-02-20 23:09:52,875 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable'. The prompt returns to 'hadoop@devopsvm:~\$' and the user enters 'S'.

```
hadoop@devopsvm:~$ sudo su - hadoop
[sudo] password for devops:
hadoop@devopsvm:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [devopsvm]
2025-02-20 23:09:52,875 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$ S
```

Рис.1 Запускаем файловую систему

Start-yarn.sh

A terminal window showing the execution of 'start-yarn.sh'. The prompt is 'hadoop@devopsvm:~\$'. The output shows 'Starting resourcemanager' and 'Starting nodemanagers'. The prompt returns to 'hadoop@devopsvm:~\$' with a cursor.

```
hadoop@devopsvm:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@devopsvm:~$
```

Рис.2 Запускаем yarn

Проверяем запущенные службы командой

jps

```
Feb 20 23:13
hadoop@devopsvm: ~
devops@devopsvm:~$ sudo su - hadoop
[sudo] password for devops:
hadoop@devopsvm:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [devopsvm]
2025-02-20 23:09:52,875 WARN util.NativeCodeLoader: Unable to load native-hadoop
  library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@devopsvm:~$ jps
3474 SecondaryNameNode
3766 ResourceManager
4536 Jps
3288 DataNode
3897 NodeManager
3118 NameNode
hadoop@devopsvm:~$
```

Рис.3 Проверка запущенных служб

Проверяем доступность запущенных систем

Переходим по ссылке для проверки запущен ли dfs по ссылке

Localhost:9870/dfshealth.html#tab-overview

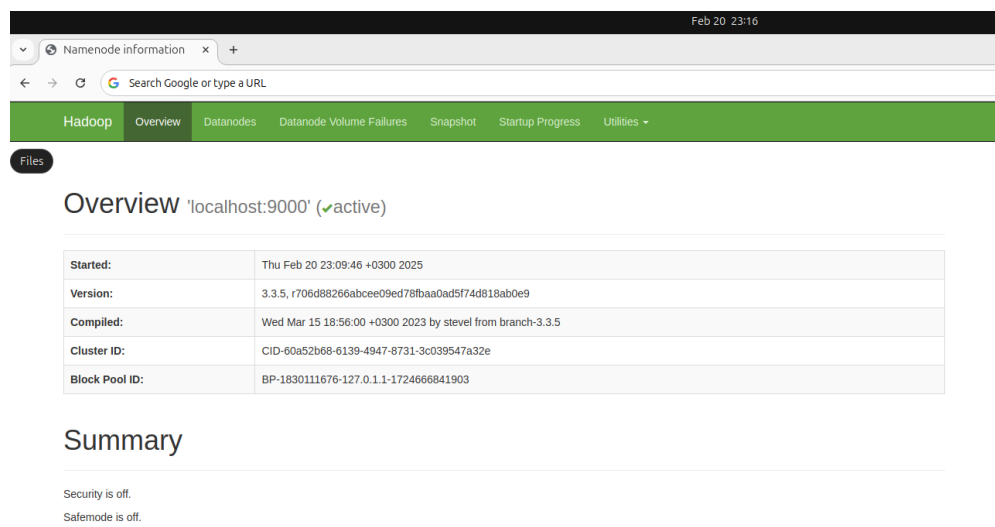


Рис.4 проверка доступности систем

Проверяем запущен ли yarn по ссылке

Localhost:8088/cluster

Feb 20 23:18

Browsing HDFS x All Applications x +

localhost:8088/cluster

Google Lens

hadoop

All Applications

Cluster

- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:4>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores
No data available in table													

Showing 0 to 0 of 0 entries

Рис.5 Проверка работы yarn

Создаем пользователя и каталог командой

Hdfs dfs -mkdir -p /lee/Hadoop/input

```
Feb 20 23:22
hadoop@devopsvm: ~
hadoop@devopsvm: ~
devops@devopsvm: ~
devops@devopsvm:~$ sudo su - hadoop
[sudo] password for devops:
hadoop@devopsvm:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [devopsvm]
2025-02-20 23:09:52,875 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@devopsvm:~$ jps
3474 SecondaryNameNode
3766 ResourceManager
4536 Jps
3288 DataNode
3897 NodeManager
3118 NameNode
hadoop@devopsvm:~$ hdfs dfs -mkdir -p /lee/hadoop/input
2025-02-20 23:20:19,590 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$
```

Рис.6 Создание пользователя

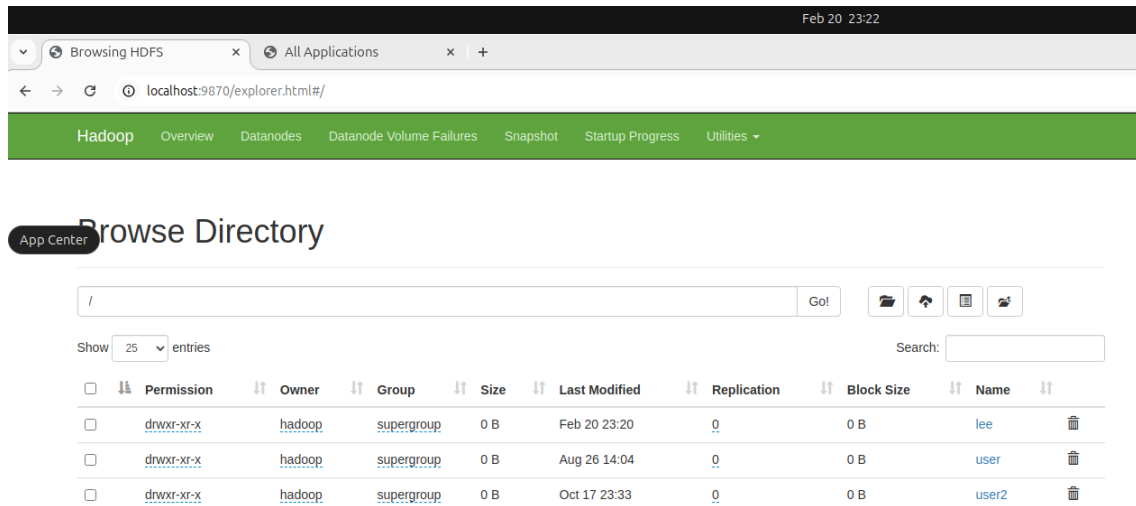


Рис.7 Проверка созданного пользователя

Скачиваем данные

wget

https://raw.githubusercontent.com/BosenkoTM/Distributed_systems/main/practice/2024/lw_01/GDP.csv

```
hadoop@devopsvm:~$ wget https://raw.githubusercontent.com/BosenkoTM/Distributed_systems/main/practice/2024/lw_01/GDP.csv
--2025-02-20 23:25:10-- https://raw.githubusercontent.com/BosenkoTM/Distributed_systems/main/practice/2024/lw_01/GDP.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.108.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 30268 (30K) [text/plain]
Saving to: 'GDP.csv.1'

GDP.csv.1      100%[=====>] 29.56K  ---KB/s   in 0.06s

2025-02-20 23:25:11 (523 KB/s) - 'GDP.csv.1' saved [30268/30268]

hadoop@devopsvm:~$
```

Рис.8 Скачиваем данные

Создаем каталог для данных

Hdfs dfs -mkdir -p /lee/Hadoop/input/economic_data

```
hadoop@devopsvm:~$ hdfs dfs -mkdir -p /lee/hadoop/input/economic_data
2025-02-20 23:29:19,690 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$
```

Рис.9 Создание каталога economic_data

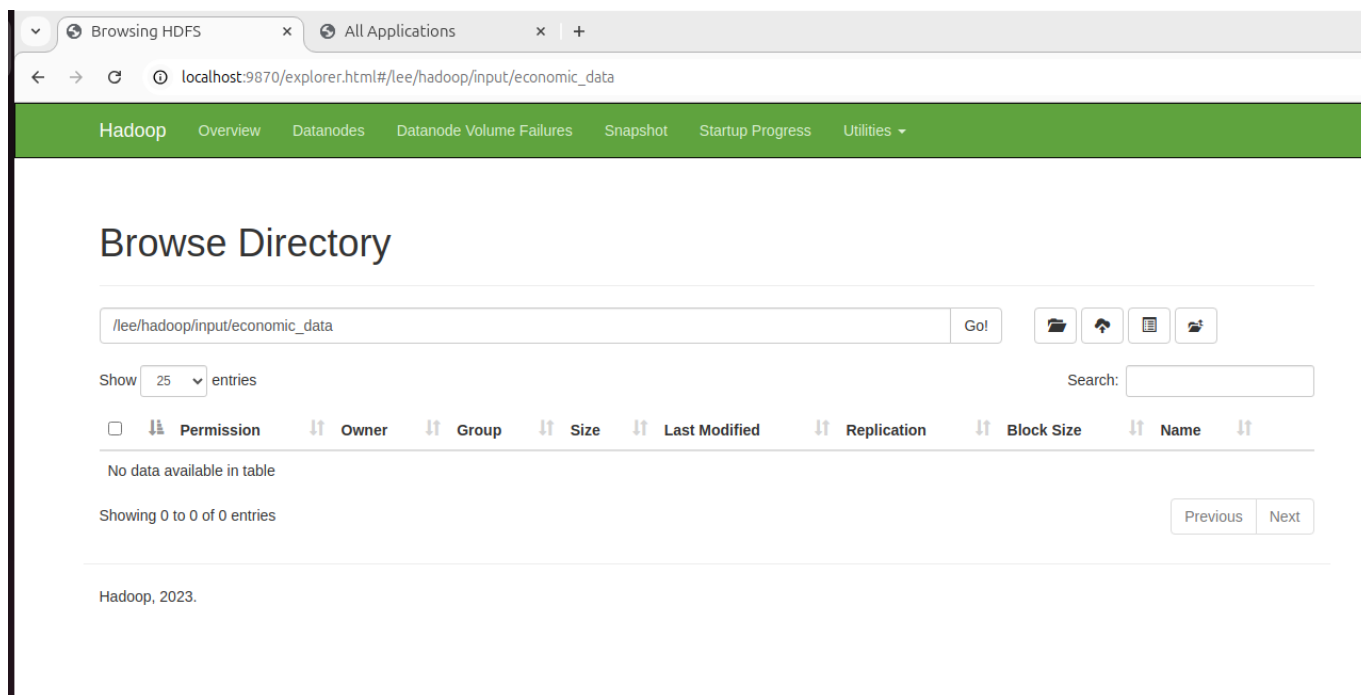


Рис.10 Проверка созданного каталога economic_data

Переносим данные в каталог

Hdfs dfs -put GDP.csv /lee/Hadoop/input/economic_data

```
hadoop@devopsvm:~$ hdfs dfs -put GDP.csv /lee/hadoop/input/economic_data
2025-02-20 23:31:22,106 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$
```

Рис.11 Перенос данных в каталог

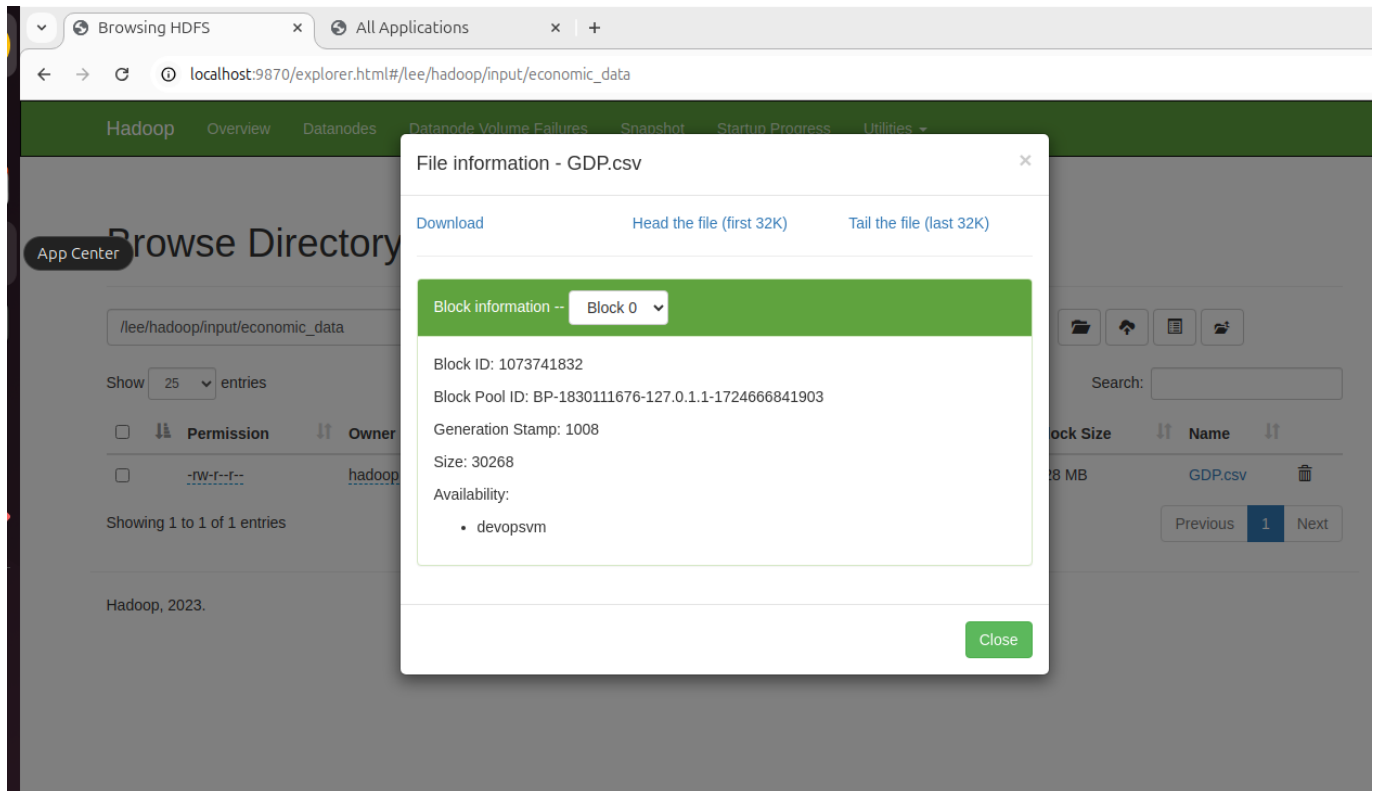


Рис.12 Проверка перенесенных данных

Задаем права доступа

hdfs dfs -chmod 777 /lee/hadoop/input/economic_data

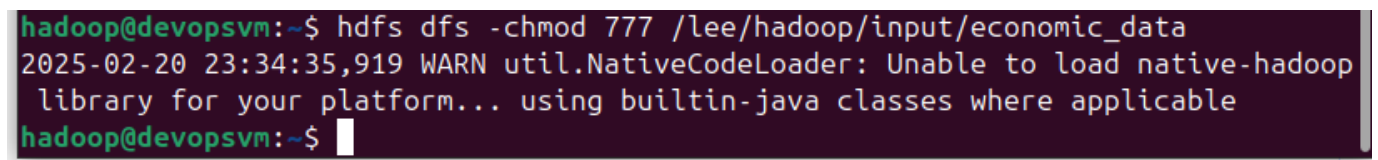


Рис.13 Устанавливаем права доступа

Обработываем данные при помощи Spark

Spark-shell

```

ress: 127.0.1.1; using 192.168.31.165 instead (on interface enp0s3)
25/02/20 23:36:04 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
25/02/20 23:36:11 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Spark context Web UI available at http://192.168.31.165:4040
Spark context available as 'sc' (master = local[*], app id = local-1740083773703
).
Spark session available as 'spark'.
Welcome to

  ____      _
 / ___|  _ \| | | |
 \___ \| |_) | |_| |
  ___) | |_) | | | |
 |____|_|_|\___|_|_|_|

version 3.4.3

Using Scala version 2.12.17 (OpenJDK 64-Bit Server VM, Java 11.0.24)
Type in expressions to have them evaluated.
Type :help for more information.

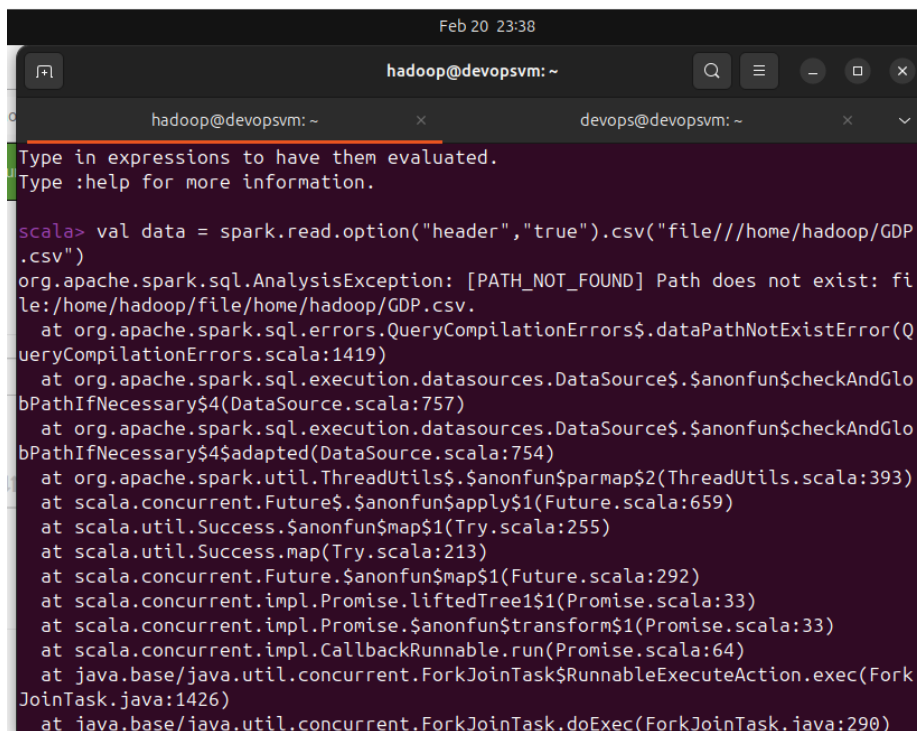
scala>

```

Рис.14 Запуск spark

Загружаем данных из hdfs

```
val data = spark.read.option("header", "true").csv("file:///home/hadoop/GDP.csv")
```



```

Feb 20 23:38
hadoop@devopsvm: ~
Type in expressions to have them evaluated.
Type :help for more information.

scala> val data = spark.read.option("header","true").csv("file:///home/hadoop/GDP
.csv")
org.apache.spark.sql.AnalysisException: [PATH_NOT_FOUND] Path does not exist: fi
le:/home/hadoop/file/home/hadoop/GDP.csv.
    at org.apache.spark.sql.errors.QueryCompilationErrors$.dataPathNotExistError(Q
ueryCompilationErrors.scala:1419)
    at org.apache.spark.sql.execution.datasources.DataSource$.anonfun$checkAndGlo
bPathIfNecessary$4(DataSource.scala:757)
    at org.apache.spark.sql.execution.datasources.DataSource$.anonfun$checkAndGlo
bPathIfNecessary$4$adapted(DataSource.scala:754)
    at org.apache.spark.util.ThreadUtils$.anonfun$parmap$2(ThreadUtils.scala:393)
    at scala.concurrent.Future$.anonfun$apply$1(Future.scala:659)
    at scala.util.Success$.anonfun$map$1(Try.scala:255)
    at scala.util.Success.map(Try.scala:213)
    at scala.concurrent.Future$.anonfun$map$1(Future.scala:292)
    at scala.concurrent.impl.Promise.liftedTree1$1(Promise.scala:33)
    at scala.concurrent.impl.Promise$.anonfun$transform$1(Promise.scala:33)
    at scala.concurrent.impl.CallbackRunnable.run(Promise.scala:64)
    at java.base/java.util.concurrent.ForkJoinTask$RunnableExecuteAction.exec(Fork
JoinTask.java:1426)
    at java.base/java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:290)

```

Рис.15 Загрузка данных

Проверка полученной схемы данных

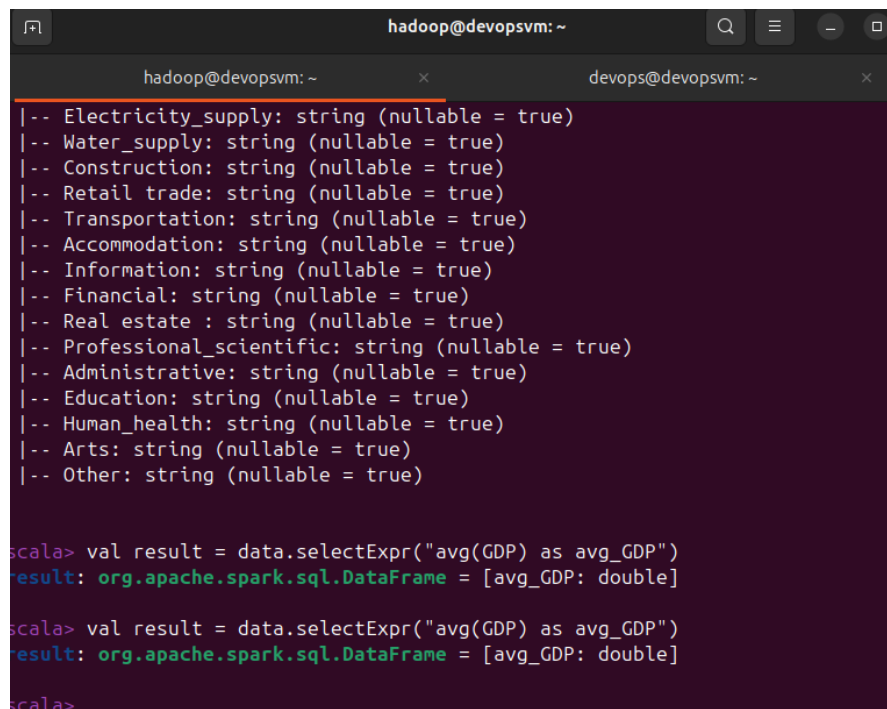
data.printSchema()

```
scala> data.printSchema()
root
 |-- Country: string (nullable = true)
 |-- Year: string (nullable = true)
 |-- GDP: string (nullable = true)
 |-- Urban_population: string (nullable = true)
 |-- Industry: string (nullable = true)
 |-- Business: string (nullable = true)
 |-- Mining: string (nullable = true)
 |-- Manufacturing: string (nullable = true)
 |-- Electricity_supply: string (nullable = true)
 |-- Water_supply: string (nullable = true)
 |-- Construction: string (nullable = true)
 |-- Retail_trade: string (nullable = true)
 |-- Transportation: string (nullable = true)
 |-- Accommodation: string (nullable = true)
 |-- Information: string (nullable = true)
 |-- Financial: string (nullable = true)
 |-- Real estate : string (nullable = true)
 |-- Professional_scientific: string (nullable = true)
 |-- Administrative: string (nullable = true)
 |-- Education: string (nullable = true)
```

Рис.12 Выводим схему

Вычисление среднего значения GDP

val result = data.selectExpr("avg(GDP) as avg_GDP")



```
hadoop@devopsvm: ~
hadoop@devopsvm: ~ x devops@devopsvm: ~ x

|-- Electricity_supply: string (nullable = true)
|-- Water_supply: string (nullable = true)
|-- Construction: string (nullable = true)
|-- Retail_trade: string (nullable = true)
|-- Transportation: string (nullable = true)
|-- Accommodation: string (nullable = true)
|-- Information: string (nullable = true)
|-- Financial: string (nullable = true)
|-- Real estate : string (nullable = true)
|-- Professional_scientific: string (nullable = true)
|-- Administrative: string (nullable = true)
|-- Education: string (nullable = true)
|-- Human_health: string (nullable = true)
|-- Arts: string (nullable = true)
|-- Other: string (nullable = true)

scala> val result = data.selectExpr("avg(GDP) as avg_GDP")
result: org.apache.spark.sql.DataFrame = [avg_GDP: double]

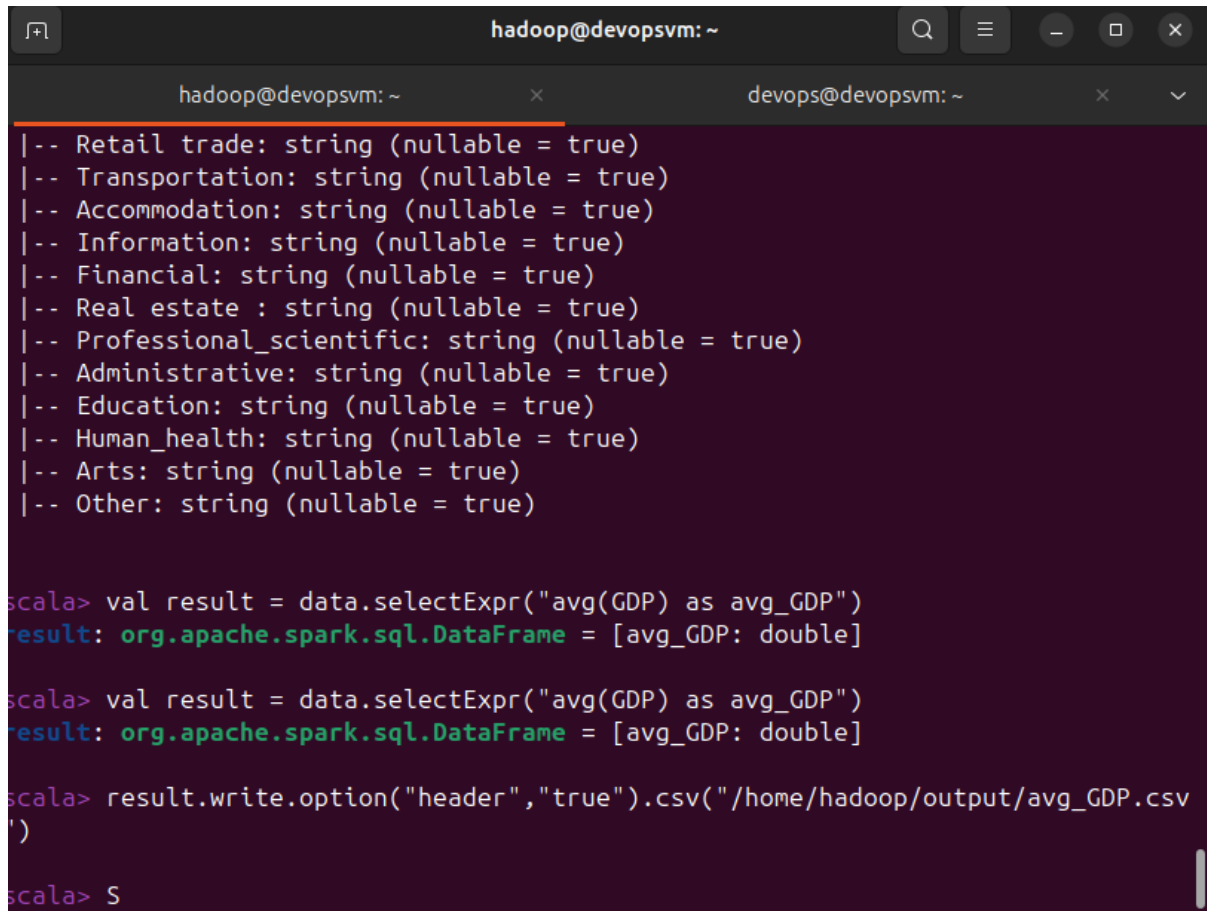
scala> val result = data.selectExpr("avg(GDP) as avg_GDP")
result: org.apache.spark.sql.DataFrame = [avg_GDP: double]

scala>
```

Рис.13 Вычисление среднего значения GDP

Сохраняем результата в CSV файл

```
result.write.option("header", "true").csv("/home/hadoop/output/avg_GDP.csv")
```

A screenshot of a terminal window with a dark background. The window title is 'hadoop@devopsvm: ~'. It shows a list of SQL-like comments for various categories: Retail trade, Transportation, Accommodation, Information, Financial, Real estate, Professional_scientific, Administrative, Education, Human_health, Arts, and Other. Below these, there are three lines of Scala code. The first two lines are identical: 'scala> val result = data.selectExpr("avg(GDP) as avg_GDP")' followed by 'result: org.apache.spark.sql.DataFrame = [avg_GDP: double]'. The third line is 'scala> result.write.option("header", "true").csv("/home/hadoop/output/avg_GDP.csv")' followed by a closing parenthesis. The prompt 'scala>' is shown at the bottom left.

```
hadoop@devopsvm: ~  
|-- Retail trade: string (nullable = true)  
|-- Transportation: string (nullable = true)  
|-- Accommodation: string (nullable = true)  
|-- Information: string (nullable = true)  
|-- Financial: string (nullable = true)  
|-- Real estate : string (nullable = true)  
|-- Professional_scientific: string (nullable = true)  
|-- Administrative: string (nullable = true)  
|-- Education: string (nullable = true)  
|-- Human_health: string (nullable = true)  
|-- Arts: string (nullable = true)  
|-- Other: string (nullable = true)  
  
scala> val result = data.selectExpr("avg(GDP) as avg_GDP")  
result: org.apache.spark.sql.DataFrame = [avg_GDP: double]  
  
scala> val result = data.selectExpr("avg(GDP) as avg_GDP")  
result: org.apache.spark.sql.DataFrame = [avg_GDP: double]  
  
scala> result.write.option("header", "true").csv("/home/hadoop/output/avg_GDP.csv")  
)  
  
scala> S
```

Рис.14 сохранение результатов

Переходим в директорию с результатами

```
cd /home/hadoop/output/avg_GDP.csv
```

A screenshot of a terminal window with a dark background. It shows two commands being executed. The first is 'hadoop@devopsvm:~\$ cd output/' which changes the directory to '/output'. The second is 'hadoop@devopsvm:~/output\$ ls' which lists the files in the current directory. The output of the 'ls' command is 'avg_GDP.csv' and 'avg_GDR.csv'. The prompt 'hadoop@devopsvm:~/output\$' is shown at the bottom left.

```
hadoop@devopsvm:~$ cd output/  
hadoop@devopsvm:~/output$ ls  
avg_GDP.csv  avg_GDR.csv  
hadoop@devopsvm:~/output$
```

Рис.15 Проверяем полученный файл

Загружаем полученный файл в HDFS

```
hdfs dfs -put /home/hadoop/output/avg_GDP.csv /user01/hadoop/input/
```

```
hadoop@devopsvm: ~/output
hadoop@devopsvm:~/output$ hdfs dfs -put /home/hadoop/output/avg_GDP.csv/avg_GDP.csv /lee/hadoop/input/
2025-02-20 23:58:55,334 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
put: `/home/hadoop/output/avg_GDP.csv/avg_GDP.csv': No such file or directory
hadoop@devopsvm:~/output$ hdfs dfs -put /home/hadoop/output/avg_GDP.csv /lee/hadoop/input/
2025-02-20 23:59:13,962 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~/output$
```

Рис.16 Загрузка файла в HDFS

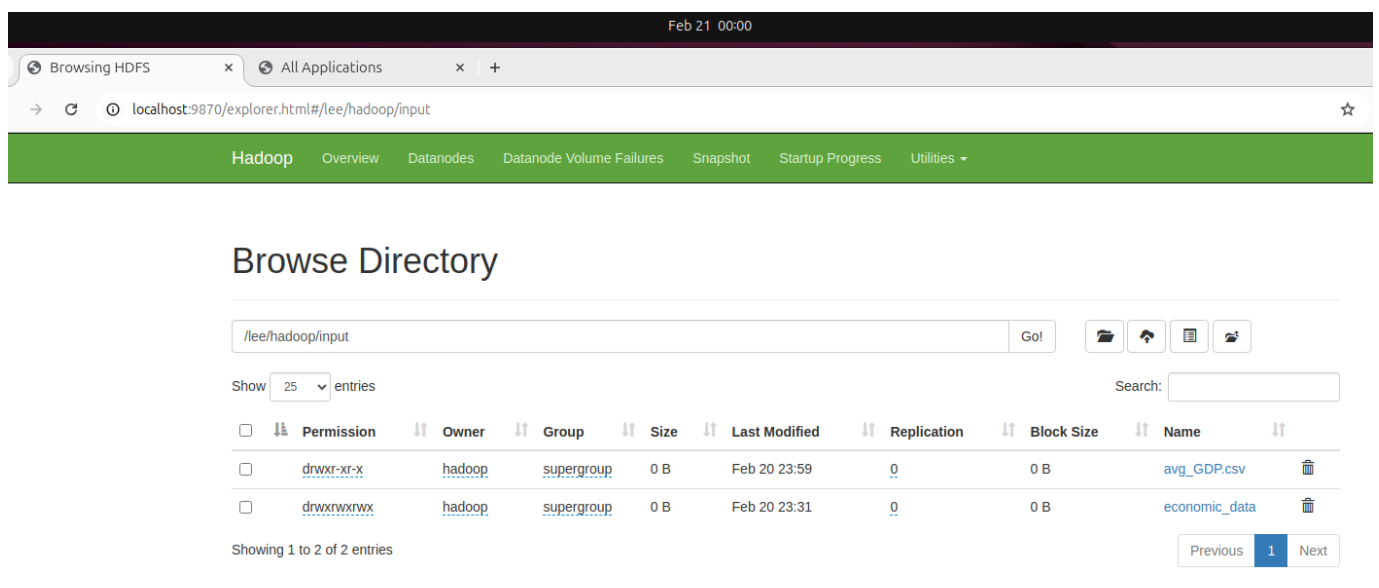
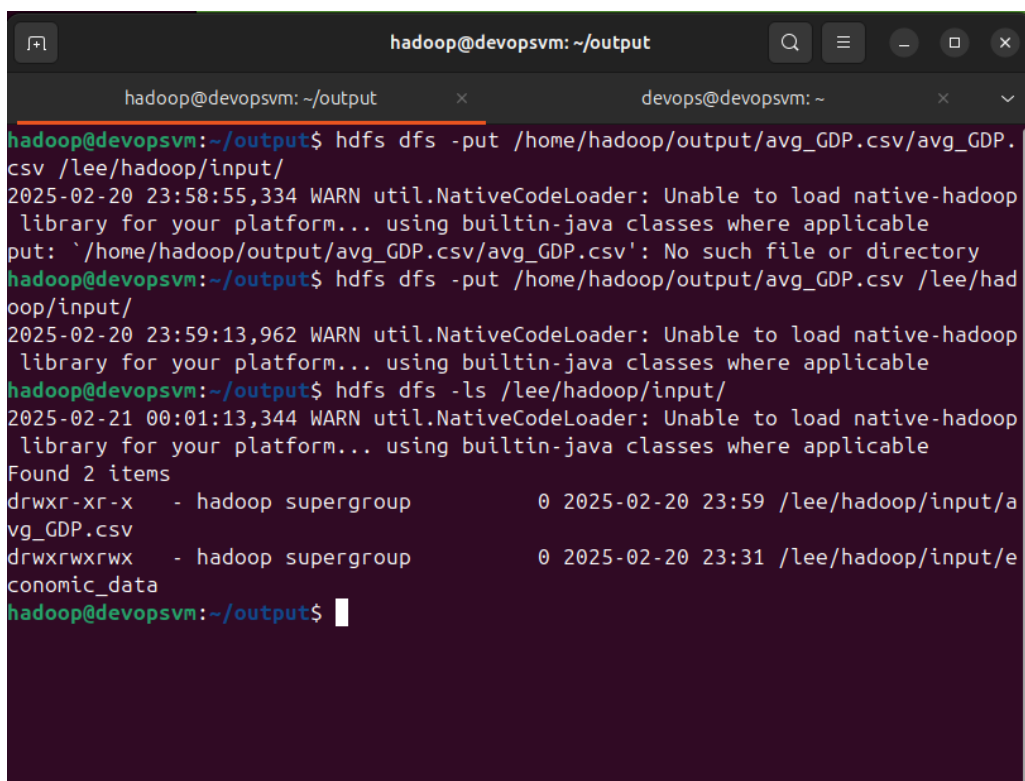


Рис.17 Проверка загруженного файла в hdfs

Проверка загрузки

hdfs dfs -ls /user01/hadoop/input/

A terminal window titled 'hadoop@devopsvm: ~/output' showing a series of Hadoop commands and their outputs. The user attempts to upload a file, receives an error, and then lists the contents of the Hadoop input directory. The output shows two files: 'avg_GDP.csv' and 'economic_data'.

```
hadoop@devopsvm:~/output$ hdfs dfs -put /home/hadoop/output/avg_GDP.csv/avg_GDP.csv /lee/hadoop/input/
2025-02-20 23:58:55,334 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
put: `/home/hadoop/output/avg_GDP.csv/avg_GDP.csv': No such file or directory
hadoop@devopsvm:~/output$ hdfs dfs -put /home/hadoop/output/avg_GDP.csv /lee/hadoop/input/
2025-02-20 23:59:13,962 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~/output$ hdfs dfs -ls /lee/hadoop/input/
2025-02-21 00:01:13,344 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x - hadoop supergroup          0 2025-02-20 23:59 /lee/hadoop/input/avg_GDP.csv
drwxrwxrwx - hadoop supergroup          0 2025-02-20 23:31 /lee/hadoop/input/economic_data
hadoop@devopsvm:~/output$
```

Рис.18 Проверка загрузки

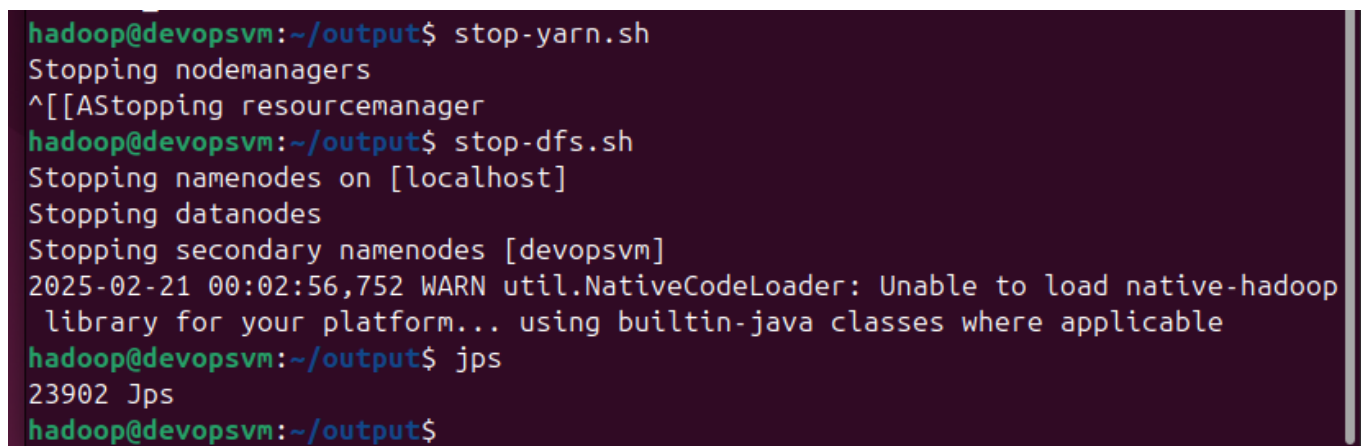
Завершение работы с Hadoop

stop-yarn.sh

stop-dfs.sh

stop-all.sh

Проверка остановки всех процессов: **jps**

A terminal window showing the execution of Hadoop stopping scripts and the 'jps' command. The output shows the stopping of nodemanagers, resourcemanager, namenodes, and datanodes. The 'jps' command shows a single process: 'Jps' with PID 23902.

```
hadoop@devopsvm:~/output$ stop-yarn.sh
Stopping nodemanagers
^[[AStopping resourcemanager
hadoop@devopsvm:~/output$ stop-dfs.sh
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [devopsvm]
2025-02-21 00:02:56,752 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~/output$ jps
23902 Jps
hadoop@devopsvm:~/output$
```

Рис.18 завершение

Задание для самостоятельной работы

Подключиться к HDFS и убедиться, что файл доступен по пути
hdfs://localhost:9000/lee/hadoop/economic_data/GDP.csv

Запускаем jupyterlab и загружаем файлы из hdfs

File_path= “hdfs://localhost:9000/lee/Hadoop/input/economic_data/GDP.csv”

df = spark.read.csv(file_path, header=True, inferSchema =True)

df.show(5)

```
[1]: # Чтение данных из HDFS
#hdfs://localhost:9000/lee/hadoop/economic_data/GDP.csv
file_path = "hdfs://localhost:9000/lee/hadoop/input/economic_data/GDP.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

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

	Country	Year	GDP	Urban_population	Industry	Business	Mining	Manufacturing	Electricity_supply	Water_supply	Construction	Retail_trade	Transportation	Accommodation	Information	Financial	Real estate	Professional_scientific	Administrative	Education	Human_health	Arts	Other
0	Austria	2010	35390	57.4	24.0	25.2	18.3	24.4	23.6	12.2	9.9	27.5	7.3	9.9	21.2	30.3	27.0	34.0	27.8	12.0	34.0	32.0	
1	Austria	2015	36140	57.72	21.8	23.4	13.7	22.7	17.6	9.3	8.2	23.3	11.6	6.4	22.4	30.3	28.0	31.0	28.0	12.9	26.2	28.3	
2	Austria	2016	36390	57.91	20.8	22.3	14.4	21.9	13.2	8.2	8.3	23.3	14.5	5.9	20.9	27.1	28.7	30.0	17.8	24.3	14.5	20.8	27.8
3	Austria	2017	36980	58.09	20.7	22.3	10.9	21.7	13.0	8.4	8.3	23.2	12.4	5.7	20.6	28.4	29.0	29.4	17.4	23.7	15.0	19.1	26.9
4	Austria	2018	37690	58.3	20.4	22.0	7.9	21.4	14.4	8.1	8.3	23.2	11.7	5.4	20.7	28.2	29.2	28.3	17.1	23.6	15.3	18.3	26.4

only showing top 5 rows

```
[1]: pandas_df = df.toPandas()
pandas_df.head()
```

```
[1]: pandas_df = df.toPandas()
pandas_df.head()
```

	Country	Year	GDP	Urban_population	Industry	Business	Mining	Manufacturing	Electricity_supply	Water_supply	...	Accommodation	Information	Financial	Real estate	Professional_scientific	Administrative	Education	Human_health	Arts	Other
0	Austria	2010	35390	57.40	24.0	25.2	18.3	24.4	23.6	12.2	...	9.9	21.2	30.3	27.0	34.0	27.8	12.0	34.0	32.0	
1	Austria	2015	36140	57.72	21.8	23.4	13.7	22.7	17.6	9.3	...	6.4	22.4	30.3	28.0	31.0	28.0	12.9	26.2	28.3	
2	Austria	2016	36390	57.91	20.8	22.3	14.4	21.9	13.2	8.2	...	5.9	20.9	27.1	28.7	30.0	17.8	24.3	14.5	20.8	27.8
3	Austria	2017	36980	58.09	20.7	22.3	10.9	21.7	13.0	8.4	...	5.7	20.6	28.4	29.0	29.4	17.4	23.7	15.0	19.1	26.9
4	Austria	2018	37690	58.30	20.4	22.0	7.9	21.4	14.4	8.1	...	5.4	20.7	28.2	29.2	28.3	17.1	23.6	15.3	18.3	26.4

5 rows × 23 columns

Рис.19 Загружаем данные из hdfs и выводим 5 строк полученных данных

Проверяем структуру данных и типы столбцов.

```
[19]:
```

	Year	GDP	Urban_population	Industry	Business	Mining	Manufacturing	Electricity_supply	Water_supply	Construction	...	Accommodation	Information	Financial	Real estate	Professional_scientific	Administrative	Education
count	267.000000	267.000000	267.000000	267.000000	267.000000	267.000000	267.000000	267.000000	267.000000	267.000000	...	267.000000	267.000000	267.000000	267.000000	267.000000	267.000000	267.000000
mean	2015.674157	26765.917603	72.079101	14.585019	16.684270	10.464794	19.172659	11.958052	2.542697	0.011610	...	10.307865	19.781273	28.531461	12.131835	18.849813	7.040449	11.697004
std	3.483933	16594.623150	12.248818	5.645432	5.053374	12.723386	6.803292	7.049383	8.338770	11.515311	...	4.792182	6.512939	7.519487	10.625204	7.752460	11.576614	6.192124
min	2010.000000	5080.000000	52.660000	0.900000	5.400000	-26.100000	1.700000	-2.000000	-24.600000	-28.300000	...	0.400000	7.300000	4.900000	-47.900000	-1.800000	-33.200000	-1.700000
max	2021.000000	71150.000000	98.120000	29.900000	30.200000	43.700000	33.600000	49.200000	18.800000	23.500000	...	27.700000	33.400000	45.100000	36.400000	36.200000	26.200000	36.000000

8 rows × 22 columns

Рис.20 Проверка структуры данных

Просматриваем типы данных столбцов

Print(pandas_df.dtypes)

```
26]: print(pandas_df.dtypes)
```

Country	object
Year	int32
GDP	int32
Urban_population	float64
Industry	float64
Business	float64
Mining	float64
Manufacturing	float64
Electricity_supply	float64
Water_supply	float64
Construction	float64
Retail trade	float64
Transportation	float64
Accommodation	float64
Information	float64
Financial	float64
Real estate	float64
Professional_scientific	float64

Рис.21 Тип данных столбцов

Проверяем данные на наличие пропущенных.

```
[27]: print(pandas_df.isna().sum())
```

Country	0
Year	0
GDP	0
Urban_population	0
Industry	0
Business	0
Mining	0
Manufacturing	0
Electricity_supply	0
Water_supply	0
Construction	0
Retail trade	0
Transportation	0
Accommodation	0
Information	0
Financial	0
Real estate	0
Professional_scientific	0

Trash

Рис.22 Проверка полученных значений

Вычисляем среднее, медиану, минимум и максимум для экономических параметров.

```
Print("mean:",pandas_df['GDP'].mean())
```

```
Print("median:",pandas_df['GDP'].median())
```

```
Print("min:",pandas_df['GDP'].min())
```

```
Print("max:",pandas_df['GDP'].max())
```

```
[28]: print("mean:",pandas_df['GDP'].mean())
      print("median:",pandas_df['GDP'].median())
      print("min:",pandas_df['GDP'].min())
      print("max:",pandas_df['GDP'].max())
```

```
mean: 26765.917602996255
median: 22970.0
min: 5080
max: 71150
```

Рис.23 Вычисляем параметры

Выявляем тенденцию

```
Trend = pandas_df.groupby('Year')['GDP'].mean()
```

```
Print (trend)
```

```
[29]: trend = pandas_df.groupby('Year')['GDP'].mean()
      print (trend)
```

```
Year
2010    25706.521739
2011    26402.000000
2012    26309.500000
2013    25521.428571
2014    23851.904762
2015    27545.238095
2016    26578.260870
2017    27256.086957
2018    27821.304348
2019    28246.250000
2020    27031.666667
2021    28374.166667
Name: GDP, dtype: float64
```

Рис.24 Выявление тенденций

Строим временные ряды, чтобы понять, как изменялась их экономика с течением времени для Италии

```
Italy_data = pandas_df[pandas_df['Country'] == 'Italy']
```

```
Italy_summary_statistics = Italy_data.describe()
```

```
plt.figure(figsize=(10,6))
```

```
plt.plot(Italy_data['Year'],Italy_data['GDP'],marker = 'o', linestyle = '-', color = 'b', label = 'GDP')
```

```
plt.title('GDP of Italy over years')
```

```
plt.xlabel('year')
```

```
plt.ylabel('GDP')
```

```
plt.legend()
```

```
plt.show()
```

Italy_summary_statistics

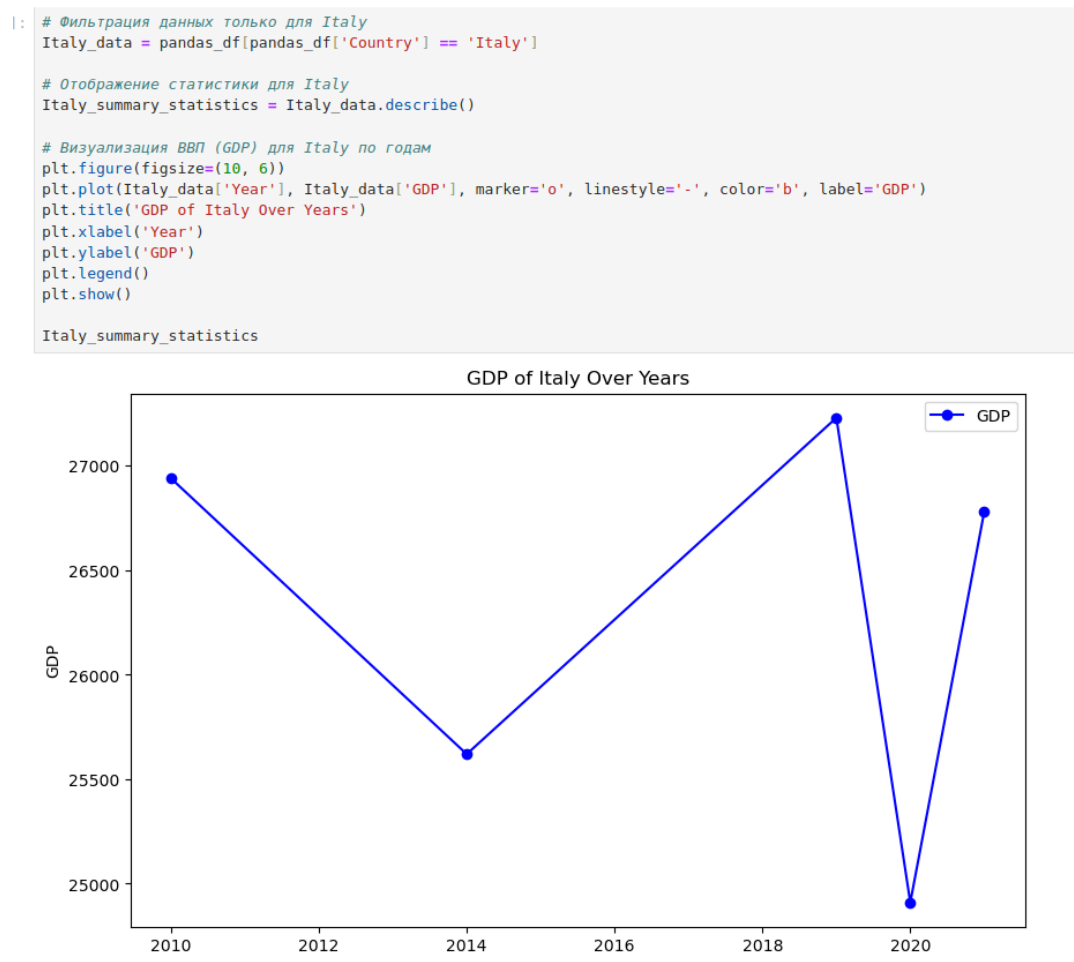


Рис.25 Построение временного ряда

Строим диаграммы для сравнения экономических показателей.


```
italy_data = pandas_df[pandas_df['Country'] == 'Italy']
labels = ['Industry', 'Business', 'Mining', 'Manufacturing', 'Electricity_supply',
'Water_supply', 'Retail trade', 'Transportation', 'Accommodation',
'Information', 'Financial', 'Professional_scientific', 'Administrative',
'Education', 'Human_health', 'Arts', 'Other']
```

```
data_mean_italy = italy_data[labels].mean().values
plt.figure(figsize=(10, 6))
```

```
plt.bar(labels, data_mean_italy, color = 'blue')
```

```
plt.xticks(rotation=90)
plt.legend()
plt.tight_layout()
plt.show()
```

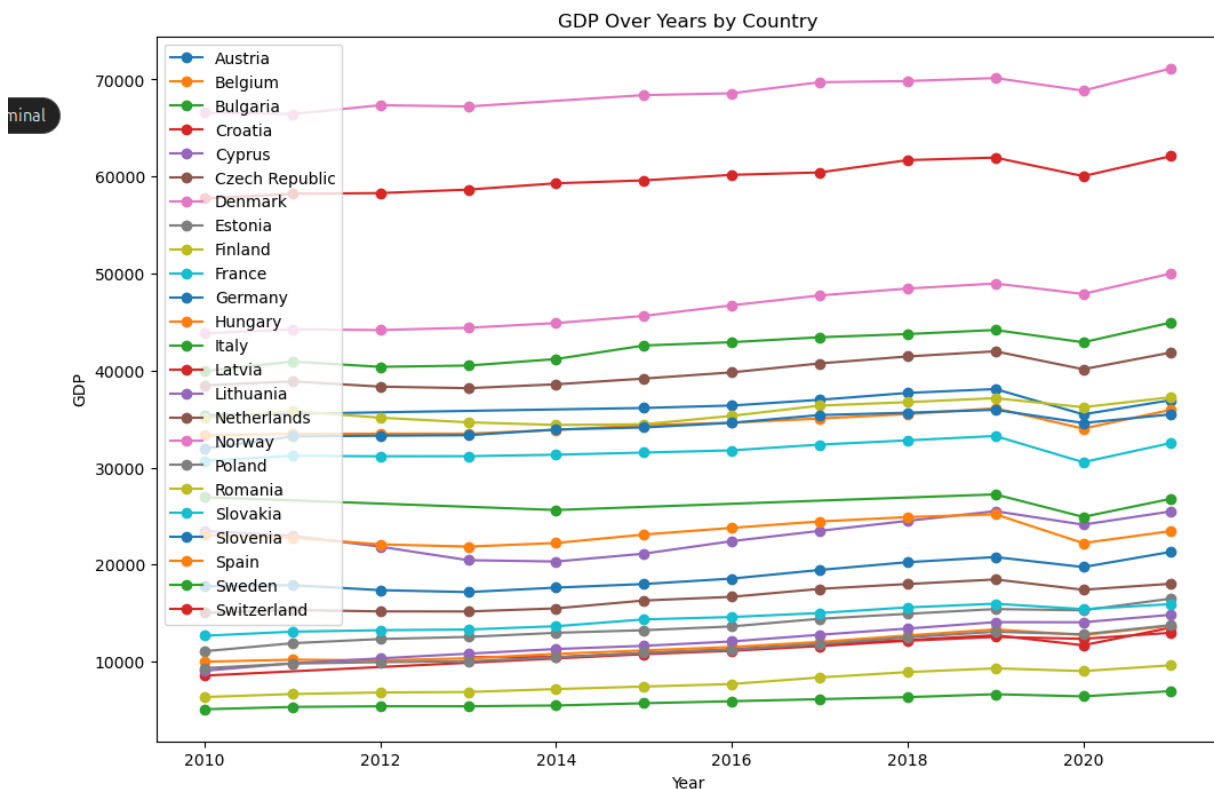


Рис.26 Диаграмма для сравнения экономических показателей

Фильтруем данные для Италии

```
italy_data = pandas_df[pandas_df['Country'] == 'Italy']
```

```
labels = ['Industry', 'Business', 'Mining', 'Manufacturing', 'Electricity_supply',
'Water_supply',
```

```
'Construction', 'Retail trade', 'Transportation', 'Accommodation', 'Information',
```

```
'Financial', 'Real estate', 'Professional_scientific', 'Administrative', 'Education',  
'Human_health', 'Arts', 'Other']
```

```
data_mean_italy = italy_data[labels].mean().values
```

```
plt.figure(figsize=(10, 6))
```

```
plt.bar(labels, data_mean_italy, color='blue')
```

```
plt.xticks(rotation=90)
```

```
plt.tight_layout()
```

```
plt.show()
```

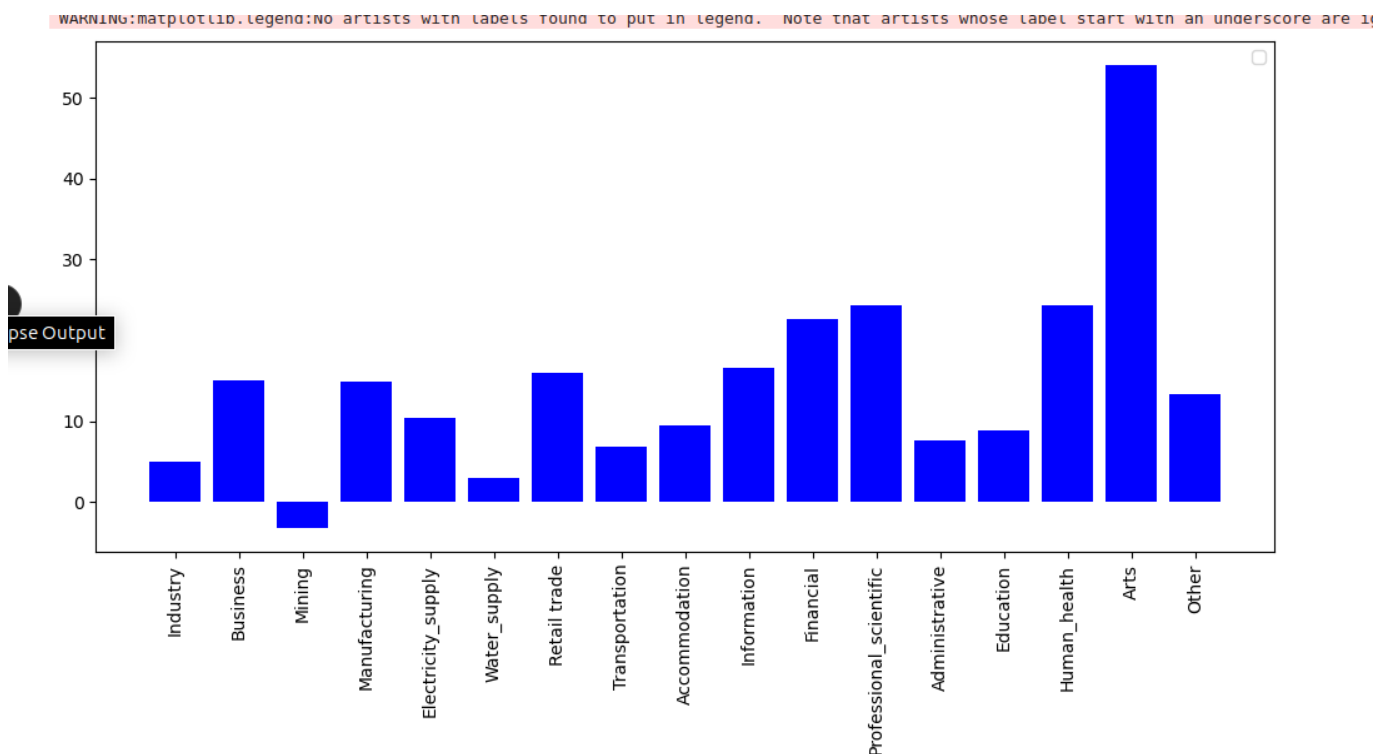


Рис.26 Диаграмма для сравнения экономических показателей для Италии

Сохраняем результат анализа и визуализации в формате CSV.

```
italy_data_spark = spark.createDataFrame(italy_data)
```

```
file_path_hdfs =
```

```
"hdfs://localhost:9000/lee/hadoop/input/economic_data/Italy_data.csv"
```

```
italy_data_spark.write.csv(file_path_hdfs, header=True, mode='overwrite')
```

Сохраняем обработанные данные обратно в HDFS

Browse Directory

/lee/hadoop/input/economic_data Go!

Show 25 entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	29.56 KB	Feb 20 23:31	1	128 MB	GDP.csv	
<input type="checkbox"/>	drwxr-xr-x	devops	supergroup	0 B	Apr 03 09:42	0	0 B	Italy_data.csv	

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2023.

Рис.27 Сохраняем обработанные данные в hdfs

/lee/hadoop/input/economic_data/Italy_data.csv Go!

Show 25 entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	devops	supergroup	0 B	Apr 03 09:42	3	128 MB	._SUCCESS	
<input type="checkbox"/>	-rw-r--r--	devops	supergroup	492 B	Apr 03 09:42	3	128 MB	part-00000-56f317e6-bb9b-4644-8e97-845c4a357f99-c000.csv	
<input type="checkbox"/>	-rw-r--r--	devops	supergroup	599 B	Apr 03 09:42	3	128 MB	part-00001-56f317e6-bb9b-4644-8e97-845c4a357f99-c000.csv	

Showing 1 to 3 of 3 entries

Previous 1 Next

Рис.28 Сохраняем обработанные данные в hdfs

Open ~ part-00000-56f317e6-bb9b-4644-8e97-845c4a357f99-c000.csv ~/Downloads

part-00000-56f317e6-bb9b-4644-8e97-845c4a357f99-c000.csv _SUCCESS part-00000-56f317e6-bb9b-4644-8e97-845c4a357f99-c000.csv

```
Country,Year,GDP,Urban_population,Industry,Business,Mining,Manufacturing,Electricity_supply,Water_supply,Construction,Retail
trade,Transportation,Accommodation,Information,Financial,Real
estate,Professional_scientific,Administrative,Education,Human_health,Arts,Other
Italy,2010,26940,68.33,5.3,15.8,-15.2,16.0,8.2,4.0,-0.6,15.5,3.1,12.5,17.9,21.7,25.2,21.1,8.4,12.6,28.5,20.4,21.3
Italy,2014,25620,69.27,6.1,17.6,6.6,17.7,11.0,4.5,9.9,17.3,6.7,8.3,18.9,22.4,27.8,25.6,10.8,7.6,28.3,61.3,13.3
```

Рис.29 Сохраняем обработанные данные в hdfs

Индивидуальное задание

Вариант 12

Исторические данные по акциям Сургутнефтегаз

https://ru.investing.com/equities/surgutneftegas_rts-historical-data

Так как ссылка не работала, нашёл в интернете данные по акциям Сургутнефтегаз в период с 03.09.2020 по 03.04.2025

Для начала создаем каталог в hdfs

Hdfs dfs -mkdir -p /lee/hadoop/input/surgutneftgaz

Showing 1 to 3 of 3 entries

Show	25	entries	Search:						
<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 20 23:59	0	0 B	avg_GDP.csv	
<input type="checkbox"/>	drwxrwxrwx	hadoop	supergroup	0 B	Apr 03 09:42	0	0 B	economic_data	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Apr 03 09:58	0	0 B	surgutneftgaz	

Previous 1 Next

Рис.30 Создание каталога surgutneftgaz в hdfs

Переносим файл из пользователя devops в пользователя Hadoop

sudo mv /home/devops/Desktop/dataset_surgutneftgaz.csv /home/hadoop

Переносим данные в каталог

Hdfs dfs -put dataset_surgutneftgaz.csv /lee/Hadoop/input/economic_data

Browse Directory

/lee/hadoop/input/surgutneftgaz

Go!

Show 25 entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	74.19 KB	Apr 03 10:16	1	128 MB	dataset_surgutneftgaz.csv	

Showing 1 to 1 of 1 entries

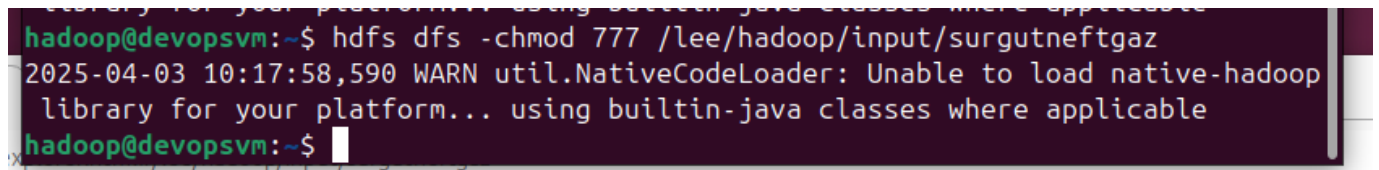
Previous 1 Next

Hadoop, 2023.

Рис.31 Переносим данные об акциях в hdfs

Устанавливаем права доступа

hdfs dfs -chmod 777 /lee/hadoop/input/surgutneftgaz



```
hadoop@devopsvm:~$ hdfs dfs -chmod 777 /lee/hadoop/input/surgutneftgaz
2025-04-03 10:17:58,590 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$
```

Рис.32 Устанавливаем права доступа

Загружаем данные из hdfs

Import pandas as pd

Import matplotlib.pyplot as plt

From pyspark.sql import SparkSession

Spark = SparkSession.builder\

.appName("Economic Data Analysis")\

.config("spark.hadoop.fs.defaultFS","hdfs://localhost:9000")\

.getOrCreate()

Spark.conf.set("spark.sql.shuffle.partition", "50")

File_path = "hdfs://localhost:9000/lee/Hadoop/input/surgutneftgaz/dataset.csv"

Df= spark.read.csv(file_path, header = True, inferSchema =True)

Df.show(5)

```
[1]: !pip install pyspark

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

[2]: import pandas as pd
import matplotlib.pyplot as plt

[3]: from pyspark.sql import SparkSession

# Создание SparkSession
spark = SparkSession.builder \
    .appName("Economic 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")

25/04/03 11:04:47 WARN Utils: Your hostname, devopsvm resolves to a loopback address: 127.0.1.1; using 192.168.31.165 instead (on interface enp0s3)
25/04/03 11:04:47 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/03 11:04:48 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

[4]: # Чтение данных из HDFS
#hdfs://localhost:9000/lee/hadoop/economic_data/GDP.csv
file_path = "hdfs://localhost:9000/lee/hadoop/input/surgutneftgaz/dataset.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Просмотр первых строк данных
df.show(5)

+-----+-----+-----+-----+-----+-----+-----+
| data | price | open | max | min | volume | change |
+-----+-----+-----+-----+-----+-----+
| 03.04.2025 | 25,310 | 25,265 | 25,495 | 25,190 | 4,96M | -0,06% |
| 02.04.2025 | 25,325 | 24,560 | 25,460 | 24,450 | 63,48M | 3,11% |
| 01.04.2025 | 24,560 | 25,535 | 25,845 | 24,410 | 49,85M | -3,82% |
| 31.03.2025 | 25,535 | 25,570 | 25,750 | 24,350 | 61,14M | 0,24% |
| 28.03.2025 | 25,475 | 26,165 | 26,285 | 25,400 | 56,68M | -2,58% |
+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

Рис.33 Загрузка данных в hdfs

Просматриваем полученные данные

```
[5]: pandas_df = df.toPandas()
pandas_df.head()
```

```
[5]:
```

	data	price	open	max	min	volume	change
0	03.04.2025	25,310	25,265	25,495	25,190	4,96M	-0,06%
1	02.04.2025	25,325	24,560	25,460	24,450	63,48M	3,11%
2	01.04.2025	24,560	25,535	25,845	24,410	49,85M	-3,82%
3	31.03.2025	25,535	25,570	25,750	24,350	61,14M	0,24%
4	28.03.2025	25,475	26,165	26,285	25,400	56,68M	-2,58%

Рис.34 Просмотр полученные данные

Фильтрация данных за 2020 год, преобразуем date из числового формата в дату

```
pandas_df["data"] = pd.to_datetime(pandas_df["data"], dayfirst=True)
```

```
df_2020 = pandas_df[pandas_df["data"].dt.year == 2020]
```

```
print(df_2020.head())
```

```
[13]: pandas_df["data"] = pd.to_datetime(pandas_df["data"], dayfirst = True)
```

```
[14]: pandas_df.head()
```

```
[14]:
```

	data	price	open	max	min	volume	change
0	2025-04-03	25,310	25,265	25,495	25,190	4,96M	-0,06%
1	2025-04-02	25,325	24,560	25,460	24,450	63,48M	3,11%
2	2025-04-01	24,560	25,535	25,845	24,410	49,85M	-3,82%
3	2025-03-31	25,535	25,570	25,750	24,350	61,14M	0,24%
4	2025-03-28	25,475	26,165	26,285	25,400	56,68M	-2,58%

```
[17]: df_2020 = pandas_df[pandas_df["data"].dt.year== 2020]
print(df_2020.head(10))
```

	data	price	open	max	min	volume	change
1057	2020-12-30	35,985	35,820	36,330	35,600	32,51M	0,83%
1058	2020-12-29	35,690	35,450	36,040	35,210	32,31M	1,22%
1059	2020-12-28	35,260	35,050	35,535	35,050	20,17M	0,57%
1060	2020-12-25	35,060	35,000	35,100	34,910	6,53M	0,30%
1061	2020-12-24	34,955	34,910	35,075	34,785	12,05M	-0,04%
1062	2020-12-23	34,970	34,895	35,055	34,705	20,27M	0,37%
1063	2020-12-22	34,840	34,350	35,055	34,115	28,29M	1,10%
1064	2020-12-21	34,460	34,990	35,070	34,040	42,18M	-1,80%
1065	2020-12-18	35,090	35,035	35,310	34,685	35,60M	0,09%
1066	2020-12-17	35,060	35,480	35,605	34,890	47,98M	-0,40%

Рис.35 Преобразование date в формат даты

Рассчитываем стандартное отклонение цены открытия

```
Df_2020["open"] = df_2020["open"].str.replace(",","").astype(float)
```

```
Std_open =df_2020["open"].std()
```

```
Print(f"standart deviation:{std_open:.2f}")
```

```
)]:
```

```
df_2020["open"] = df_2020["open"].str.replace(",","").astype(float)
```

```
std_open = df_2020["open"].std()
```

```
print(f"Standard deviation: {std_open:.2f}")
```

```
Standard deviation: 1093.07
```

Рис.36 Стандартное отклонение цены открытия

Тренд-анализ

```
df_2020["Days"] = (df_2020["data"] - df_2020["data"].min()).dt.days
slope, intercept, r_value, p_value, std_err = linregress(df_2020["Days"],
df_2020["open"])
```

```

df_2020["Trend"] = intercept + slope * df_2020["Days"]
plt.figure(figsize=(10, 5))
plt.plot(df_2020["data"], df_2020["open"], label="Цена открытия",
linestyle="dotted", color="blue")
plt.plot(df_2020["data"], df_2020["Trend"], label="Тренд (Линейная
регрессия)", color="red")
plt.xlabel("data")
plt.ylabel("price open")
plt.title("Тренд цены открытия за 2020 год")
plt.legend()
plt.show()

```



Рис.37 Тренд анализ

Расчет волатильности и анализ динамики цен открытия

```
Pandas_df["open"] = pandas_df["open"].str.replace(",", ".").astype(float)
```

```
Volatility = pandas_df["open"].std()
```

```
Print(volatility)
```

```

]: pandas_df["open"] = pandas_df["open"].str.replace(",", ".").astype(float)
volatility = pandas_df["open"].std()
print(volatility)

```

```
5.4689643590088295
```

Рис.38 Расчет волатильности

Анализ динамики цен открытия

```
pandas_df["price change"] = pandas_df["open"].pct_change() * 100
plt.figure(figsize=(12, 8))
plt.plot(pandas_df["data"], pandas_df["price change"], label="price change open", color="green")
plt.xlabel('data')
plt.ylabel('price change %')
plt.legend()
plt.show()
```

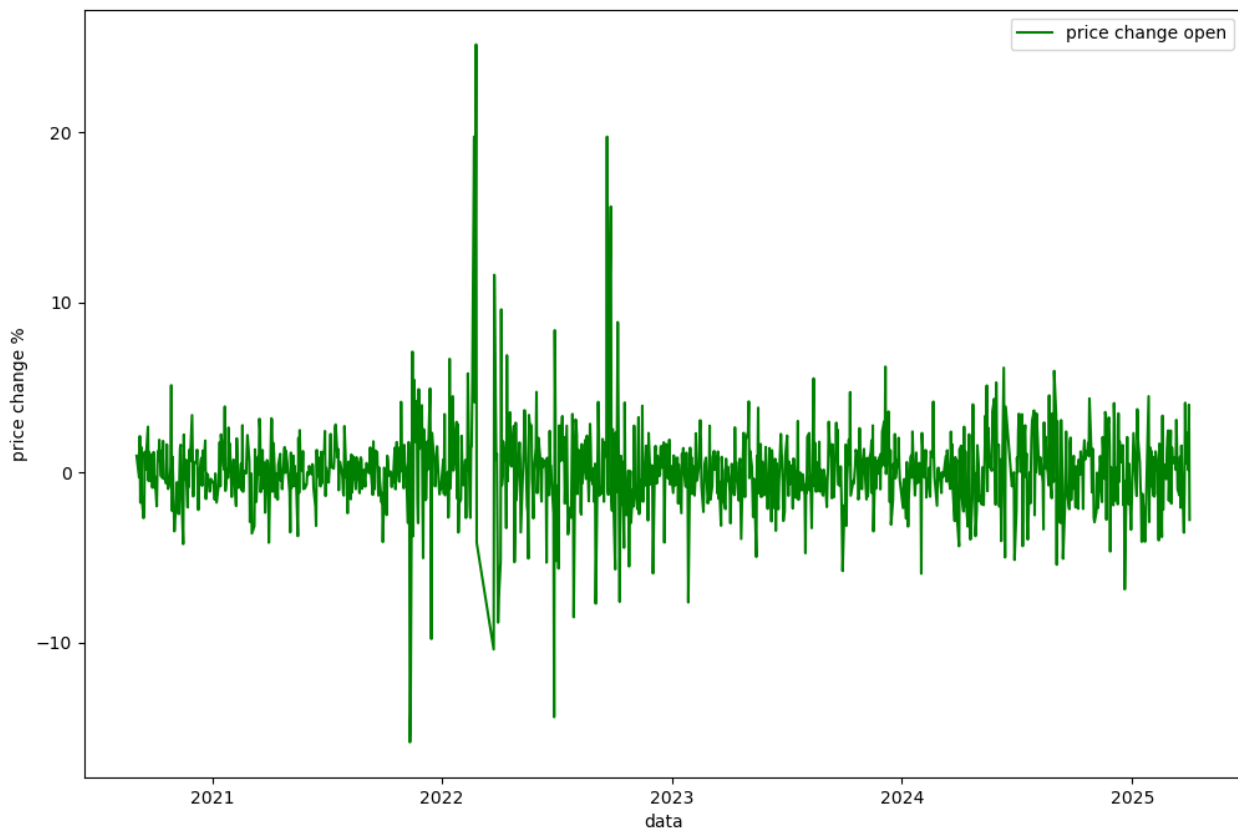


Рис.39 Анализ динамики цен открытия

Заключение

В ходе проделанной лабораторной работы, были изучены основные операции и функциональные возможности системы, что позволило понять принципы работы с данными и распределенными вычислениями, также было выполнено задание по вариантам.