

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

Инструменты хранения и анализа больших данных

Смоляков Руслан Игоревич БД-241м

**Практическая работа 1.2. Обработка данных с использованием Apache
Spark и Python (PySpark)**

Вариант 23

Направление подготовки/специальность
38.04.05 - Бизнес-информатика
Бизнес-аналитика и большие данные
(очная форма обучения)

Руководитель дисциплины:
Босенко Т.М., доцент департамента
информатики, управления и технологий,
кандидат технических наук

Москва
2025

Введение

Цель: освоение основ работы с Apache Spark и его интеграцией с Python через библиотеку PySpark. Студенты научатся обрабатывать большие объемы данных, используя распределенные вычисления, а также научатся применять базовые операции с RDD (Resilient Distributed Datasets) и DataFrame, работать с SQL-запросами в Spark SQL, а также визуализировать результаты обработки данных.

Задачи:

- 1. Установить Apache Spark и PySpark.**
Настроить рабочую среду для использования PySpark в Python, установить необходимые зависимости и настроить Spark на локальном компьютере или через облачную платформу.
- 2. Загрузка данных и их предварительная обработка.**
Скачать или подготовить исходные данные для анализа (например, текстовые файлы или CSV). Загружать данные в Spark через RDD или DataFrame, выполнить предварительную обработку: очистка данных, фильтрация, преобразования.
- 3. Применение операций с RDD и DataFrame.**
Научиться работать с RDD и DataFrame, выполнять такие операции как map, filter, reduce, groupBy, join и другие стандартные операции для обработки данных в распределенной среде.
- 4. Применение SQL-запросов через Spark SQL.**
Использование SQL-запросов в Spark для извлечения и агрегации данных, создание временных таблиц и выполнение сложных запросов для анализа данных.
- 5. Визуализация результатов анализа данных.**
Визуализировать полученные результаты с помощью библиотеки Python для визуализации данных (например, matplotlib или seaborn). Построить графики для лучшего представления результатов.
- 6. Подготовка отчета.**
Оформить отчет по выполненной практике, в котором будет описан процесс выполнения работы, анализ полученных результатов и выводы. Включить ссылки на репозиторий и прикрепить сам отчет в формате PDF или Markdown.

Вариант 23

Задание 1 (Spark + Hadoop): Анализ качества: загрузить quality.csv в HDFS, выявить проблемные товары

Задание 2 (Spark Local + SQL): SQL-анализ: определить процент брака по производителям

Задание 3 (Визуализация): Построить диаграмму качества продукции

Перейдем на пользователя «hadoop» и запустим Hadoop:

```
devops@devopsvm:~$ sudo su - hadoop
[sudo] password for devops:
hadoop@devopsvm:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [devopsvm]
2025-05-03 03:45:37,064 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
3507 SecondaryNameNode
4292 Jps
3802 ResourceManager
3066 NameNode
3307 DataNode
3933 NodeManager
hadoop@devopsvm:~$
```

Browse Directory

Show

25

 entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 18 21:21	0	0 B	admin01	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Apr 29 00:49	0	0 B	user	<input type="checkbox"/>

← → ↺ localhost:8088/cluster

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
0	0	0	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes
1	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type
Capacity Scheduler	[memory-mb (unit=M), vcores]

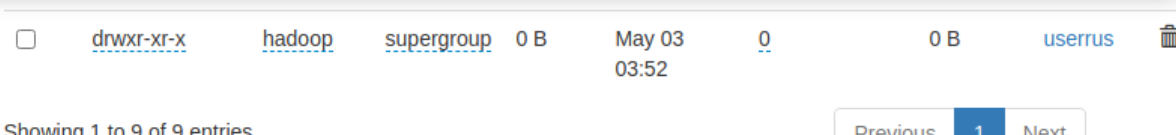
Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	Start
----	------	------	------------------	------------------	-------	----------------------	-------

Showing 0 to 0 of 0 entries

Создали директорию и сразу проверяем:

```
hadoop@devopsvm:~$ hdfs dfs -mkdir -p /usererus/sparkdir
2025-05-03 03:52:07,144 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$ S
```



Showing 1 to 9 of 9 entries

Previous 1 Next

1. Задание 1 (Spark + Hadoop): Анализ качества: загрузить quality.csv в HDFS, выявить проблемные товары

```
hadoop@devopsvm:~$ hdfs dfs -put /home/hadoop/quality.csv /usererus/
2025-05-03 04:02:53,933 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

Откроем блокнот, проверим подключение и выведем первые 5 строк из датасета:

```
from pyspark.sql import SparkSession

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

# Чтение данных из HDFS
file_path = "hdfs://localhost:9000/usererus/quality.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

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

```
+-----+-----+-----+-----+
|product_id| manufacturer|  quality|batch_id|
+-----+-----+-----+-----+
|          1|ManufacturerS|    Good|    176|
|          2|ManufacturerD|Defective|    196|
|          3|ManufacturerI|Defective|    178|
|          4|ManufacturerS|    Good|    132|
|          5|ManufacturerM|    Good|    152|
+-----+-----+-----+-----+
only showing top 5 rows
```

Результат выполнения задания:

```
24]: from pyspark.sql.functions import col

# Чтение данных из HDFS
file_path = "hdfs://localhost:9000/userrus/quality.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Фильтрация проблемных товаров (качество = "Defective")
problematic_products = df.filter(col("quality") == "Defective")

# Вывод проблемных товаров
problematic_products.show()

+-----+-----+-----+-----+
|product_id| manufacturer|  quality|batch_id|
+-----+-----+-----+-----+
|         2|ManufacturerD|Defective|   196|
|         3|ManufacturerI|Defective|   178|
|         8|ManufacturerM|Defective|   156|
|        10|ManufacturerI|Defective|   132|
|        12|ManufacturerG|Defective|   130|
|        13|ManufacturerU|Defective|   155|
|        14|ManufacturerH|Defective|   181|
|        15|ManufacturerL|Defective|   103|
|        17|ManufacturerH|Defective|   196|
|        18|ManufacturerV|Defective|   122|
|        21|ManufacturerO|Defective|   106|
|        22|ManufacturerV|Defective|   167|
|        23|ManufacturerJ|Defective|   186|
|        24|ManufacturerH|Defective|   176|
|        25|ManufacturerA|Defective|   122|
|        26|ManufacturerM|Defective|   183|
|        30|ManufacturerP|Defective|   176|
|        32|ManufacturerL|Defective|   174|
|        33|ManufacturerP|Defective|   176|
|        34|ManufacturerG|Defective|   164|
+-----+-----+-----+-----+
only showing top 20 rows
```

2. Задание 2 (Spark Local + SQL): SQL-анализ: определить процент брака по производителям

```
[27]: from pyspark.sql.functions import col, when, sum as spark_sum

# Чтение данных из HDFS
file_path = "hdfs://localhost:9000/userrus/quality.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Группировка данных по производителю и расчет процента брака
defect_percentage_df = df.groupBy("manufacturer") \
    .agg(
        spark_sum(when(col("quality") == "Defective", 1).otherwise(0)).alias("defective_count"),
        spark_sum(when(col("quality").isNull(), 1).otherwise(0)).alias("total_count")
    ) \
    .withColumn("defect_percentage", (col("defective_count") / col("total_count") * 100).cast("double"))

# Вывод результатов
defect_percentage_df.show()

+-----+-----+-----+-----+
| manufacturer|defective_count|total_count| defect_percentage|
+-----+-----+-----+-----+
|ManufacturerP|         22|        35|62.857142857142854|
|ManufacturerK|         21|        41| 51.21951219512195|
|ManufacturerM|         22|        41| 53.65853658536586|
|ManufacturerO|         14|        37| 37.83783783783784|
|ManufacturerT|         24|        45|53.333333333333336|
|ManufacturerR|         24|        43| 55.81395348837209|
|ManufacturerS|         24|        43| 55.81395348837209|
|ManufacturerH|         21|        35|        60.0|
|ManufacturerI|         23|        50|        46.0|
|ManufacturerE|         17|        34|        50.0|
|ManufacturerZ|         13|        28| 46.42857142857143|
|ManufacturerU|         17|        40|        42.5|
|ManufacturerD|         24|        48|        50.0|
|ManufacturerC|         18|        46|39.130434782608695|
|ManufacturerG|         19|        34| 55.88235294117647|
|ManufacturerV|         17|        31| 54.83870967741935|
|ManufacturerX|         17|        37| 45.94594594594595|
|ManufacturerA|         10|        28|35.714285714285715|
|ManufacturerO|         20|        38| 52.63157894736842|
|ManufacturerJ|         19|        35|54.285714285714285|
+-----+-----+-----+-----+
only showing top 20 rows
```

3. Задание 3 (Визуализация): Построить диаграмму качества продукции

```
import pandas as pd
import matplotlib.pyplot as plt
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, sum as spark_sum

# Чтение данных из HDFS (исходный датасет)
file_path = "hdfs://localhost:9000/usererrus/quality.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Группировка данных по производителю и расчет процента брака
defect_percentage_df = df.groupBy("manufacturer") \
    .agg(
        spark_sum(when(col("quality") == "Defective", 1).otherwise(0)).alias("defective_count"),
        spark_sum(when(col("quality").isNull(), 1).otherwise(0)).alias("total_count")
    ) \
    .withColumn("defects_percentage", (col("defective_count") / col("total_count") * 100).cast("double"))

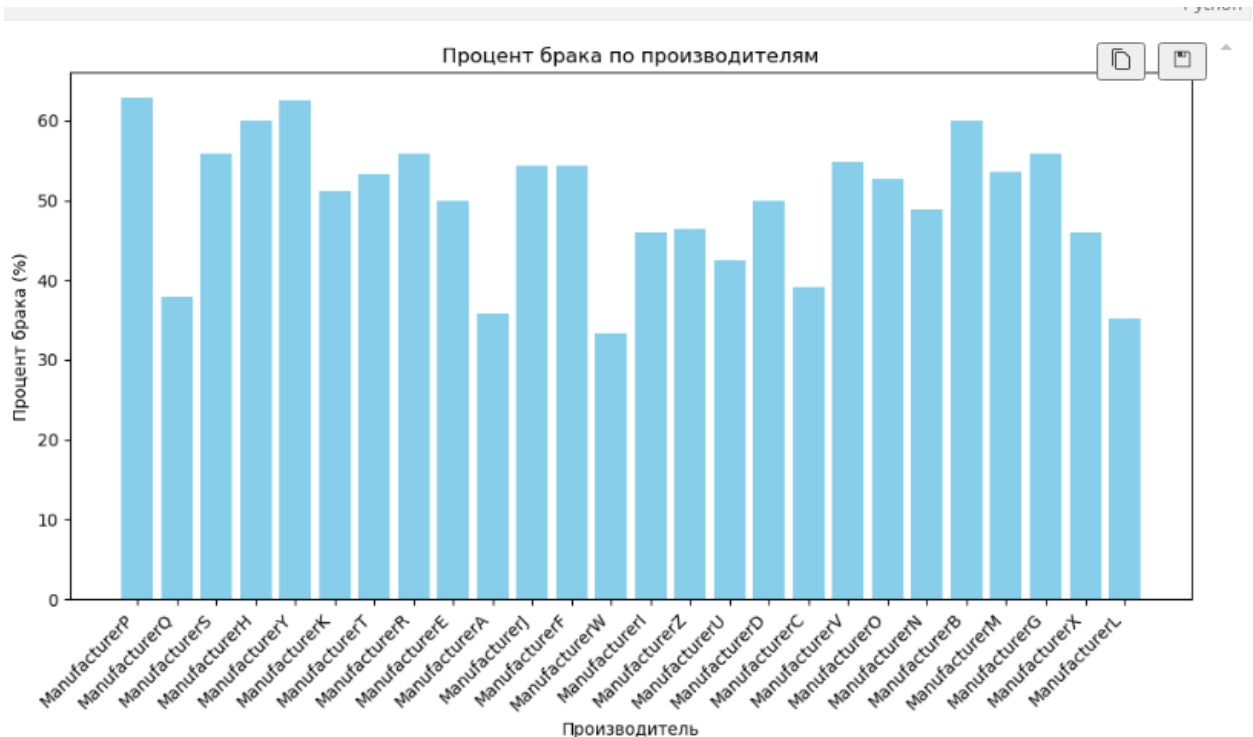
# Преобразование Spark DataFrame в Pandas DataFrame
pandas_df = defect_percentage_df.toPandas()

# Построение диаграммы
plt.figure(figsize=(10, 6))
plt.bar(pandas_df['manufacturer'], pandas_df['defects_percentage'], color='skyblue')
plt.xlabel('Производитель')
plt.ylabel('Процент брака (%)')
plt.title('Процент брака по производителям')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Сохранение диаграммы в файл
plt.savefig('defect_percentage_chart.png')

# Отображение диаграммы
plt.show()
```

Вывод:



Вывод:

В ходе выполнения практической работы были успешно освоены основы работы с Apache Spark и его интеграция с Python через библиотеку PySpark. В процессе выполнения заданий были приобретены навыки обработки больших данных с использованием распределенных вычислений, включая работу с RDD и DataFrame, выполнение SQL-запросов в Spark SQL, а также визуализацию результатов анализа данных.