Выполнил: Бойко Илья Группа: ПИН-б-о-22-1

Лабораторная работа №4: Обработка структурированных данных с помощью Spark SQL

1. Цель работы Освоить базовые принципы работы с низкоуровневой абстракцией Apache Spark — RDD (Resilient Distributed Dataset). Получить практические навыки создания RDD, применения основных трансформаций (transformations) и действий (actions), а также построения и выполнения последовательностей операций (DAG).

2. Используемый стек технологий

Язык программирования: Python (PySpark)

Фреймворк: Apache Spark

Язык программирования: Python

Инструментарий:Jupyter Lab / Jupyter Notebook

Данные: Структурированные данные в форматах JSON/CSV

3. Теоретические сведения

Spark SQL — это модуль в Spark для работы со структурированными данными. Он предоставляет два основных API:

1. **DataFrames:** Распределенная коллекция данных, организованная в именованные столбцы. Концептуально эквивалентна таблице в реляционной БД или dataframe в R/Pandas, но с оптимизациями под капотом (Catalyst Optimizer, Tungsten Execution Engine).
2. **Datasets:** Типизированная версия DataFrame (доступна в Java/Scala, в Python эквивалентна DataFrame).

Преимущества DataFrame над RDD:

- Высокая производительность за счет оптимизации запросов
- Возможность использования SQL-синтаксиса
- Встроенная поддержка схемы данных
- Интеграция с различными источниками данных

4. **Ход выполнения работы**

Часть 1: Создание сессии Spark и загрузка данных

4.1 Создание Spark Session

from pyspark.sql import SparkSession from pyspark.sql.functions import col, avg, count, desc, round, sum as _sum import matplotlib.pyplot as plt import seaborn as sns

spark = SparkSession.builder
.appName("Lab4_Structured_Data_Analysis")
.config("spark.sql.repl.eagerEval.enabled", True)
.config("spark.sql.repl.eagerEval.maxNumRows", 10)
.getOrCreate()

4.2 Создание тестовых данных

- employees.csv

employees_data = """id,name,department,salary 1,Ivanov,IT,75000 2,Petrov,HR,50000
3,Sidorov,IT,80000 4,Smirnov,Sales,60000 5,Kuznetsov,HR,55000 6,Nikitin,IT,85000
7,Orlova,Sales,65000 8,Fedorov,IT,78000 9,Morozova,HR,52000 10,Pavlova,Sales,62000"""

with open('employees.csv', 'w', encoding='utf-8') as f: f.write(employees_data)

- departments.json

departments_data = """{"dep_id": "IT", "dep_name": "Information Technology", "location": "Floor 3"}
{"dep_id": "HR", "dep_name": "Human Resources", "location": "Floor 2"} {"dep_id": "Sales",
"dep_name": "Sales Department", "location": "Floor 1"}"""

- JSON файл

with open('departments.json', 'w', encoding='utf-8') as f: f.write(departments_data)

4.3 Загрузка данных в Spark DataFrame

employees_df = spark.read
.option("header", "true")
.option("inferSchema", "true")
.csv("employees.csv")

departments_df = spark.read
.option("multiline", "false")
.json("departments.json")

Результат выполнения: text СХЕМА ДАННЫХ СОТРУДНИКОВ (employees_df):

root |-- id: integer (nullable = true) |-- name: string (nullable = true) |-- department: string (nullable = true)
|-- salary: integer (nullable = true)

СХЕМА ДАННЫХ ДЕПАРТАМЕНТОВ (departments_df):

root |-- dep_id: string (nullable = true) |-- dep_name: string (nullable = true) |-- location: string (nullable = true)

Часть 2: Работа с DataFrame API 4.1 Базовые операции 4.1.1 Показ первых записей

employees_df.show(5, truncate=False)

departments_df.show(5, truncate=False)

Результат выполнения:

text

ПЕРВЫЕ 5 ЗАПИСЕЙ СОТРУДНИКОВ:

id	name	department	salary
1	Ivanov	IT	75000
2	Petrov	HR	50000
3	Sidorov	IT	80000
4	Smirnov	Sales	60000
5	Kuznetsov	HR	55000

ПЕРВЫЕ 5 ЗАПИСЕЙ ДЕПАРТАМЕНТОВ:

dep_id	dep_name	location
IT	Information Technology	Floor 3
HR	Human Resources	Floor 2
Sales	Sales Department	Floor 1

4.1.2 Фильтрация сотрудников с зарплатой выше 50000

high_salary_employees = employees_df.filter(col("salary") > 50000)
high_salary_employees.show(truncate=False)

print(f"\n Количество сотрудников с зарплатой > 50,000: {high_salary_employees.count()}")
Результат выполнения:

text СОТРУДНИКИ С ЗАРПЛАТОЙ ВЫШЕ 50,000:

id	name	department	salary
1	Ivanov	IT	75000
3	Sidorov	IT	80000
4	Smirnov	Sales	60000
5	Kuznetsov	HR	55000
6	Nikitin	IT	85000
7	Orlova	Sales	65000
8	Fedorov	IT	78000
9	Morozova	HR	52000
10	Pavlova	Sales	62000

Количество сотрудников с зарплатой > 50,000: 9

4.1.3 Группировка по департаментам и расчет средней зарплаты

```
avg_salary_by_dept = employees_df.groupBy("department")
.agg( round(avg("salary"), 2).alias("avg_salary"), count("*").alias("employee_count"),
_sum("salary").alias("total_salary") )
.orderBy(desc("avg_salary"))

avg_salary_by_dept.show(truncate=False)
```

Результат выполнения:

text

СРЕДНЯЯ ЗАРПЛАТА ПО ДЕПАРТАМЕНТАМ:

department	avg_salary	employee_count	total_salary
IT	79500.0	4	318000
Sales	62333.33	3	187000
HR	52333.33	3	157000

4.1.4 Сортировка по убыванию средней зарплаты

```
ranking_df = avg_salary_by_dept.select( "department",
col("avg_salary").cast("int").alias("avg_salary_int"), "employee_count" )

ranking_df.show(truncate=False)
```

Визуализация

```
dept_data = ranking_df.collect()

dept_names = [row["department"] for row in dept_data] avg_salaries = [row["avg_salary_int"] for row in
dept_data] emp_counts = [row["employee_count"] for row in dept_data]

plt.figure(figsize=(10, 6)) bars = plt.bar(dept_names, avg_salaries, color=['#2E86AB', '#A23B72',
'#F18F01']) plt.title('Средняя зарплата по департаментам', fontsize=14, fontweight='bold')
plt.xlabel('Департамент', fontsize=12) plt.ylabel('Средняя зарплата', fontsize=12) plt.grid(axis='y',
alpha=0.3)

for bar, salary, count in zip(bars, avg_salaries, emp_counts): height = bar.get_height()
plt.text(bar.get_x() + bar.get_width()/2., height + 500, f'${salary:,}\n({count} чел.)', ha='center',
va='bottom', fontsize=10)

plt.tight_layout() plt.show()
```