Федеральное государственное бюджетное образовательное учреждение высшего образования   
«Саратовский государственный технический университет имени Ю. А. Гагарина»

Кафедра прикладных информационных технологий

**Практическая работа по курсу**

**Анализ больших данных на тему:**

**«Spark SQL»**

Выполнил студент б1-ИФСТ-41,

Мокляк Никита Вячеславович

Проверил преподаватель

Кузьмин Алексей Константинович

Саратов, 2022

Цель работы

* Научиться устанавливать Spark совместно с Hadoop HDFS
* Обучиться основам работы и принципам SparkSQL и сохранять получившиеся результаты в разных форматах

Краткое описание технологий

**Apache Hadoop** is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly.

Hadoop consists of four main modules:

* Hadoop Distributed File System (HDFS) – A distributed file system that runs on standard or low-end hardware. HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets.
* Yet Another Resource Negotiator (YARN) – Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.
* MapReduce – A framework that helps programs do the parallel computation on data. The map task takes input data and converts it into a dataset that can be computed in key value pairs. The output of the map task is consumed by reduce tasks to aggregate output and provide the desired result.
* Hadoop Common – Provides common Java libraries that can be used across all modules.

**Docker** is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

**Apache Spark** — фреймворк с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop.

## Задание

Установите Spark локально на операционную систему.

Запустите интепретатор Scala/Python и выполнить простой код (например, подсчет количества слов в тексте)

При помощи встроенных средств выведите информацию о Spark в консоль

Выведите web-страницу с информацией о Spark и его текущей загрузке

Установите Spark на кластер Hadoop, созданный в предыдущей практической работе

При помощи встроенных средств выведите информацию о Spark в консоль

Выведите web-страницу с информцией о Spark и его текущей загрузке

Сконфигуруйте и подготовьте к работе Spark SQL

Выберите предметную область и продумайте, как минимум, три таблицы, в которых будет содержаться информация о связанных между собой сущностях. Например, это может быть схема "сотрудники-отделы-должности" или "товары-категории-тары" и т.п.

Подготовьте JSON-файл, содержащий массив объектов для одной из таблиц. Каждый объект должен включать в себя порядка 3-4 полей разных типов. Всего объектов в массиве должно быть порядка 10

Подготовьте CSV-файл, содержащий информацию об объектах для одной из таблиц. При этом самостоятельно выберите символы, которые будут служить разделителями полей и строк.

Объявите коллекцию, включающую в себя информацию об объектах для одной из таблиц. Всего должно быть более 5 объектов, с, как минимум, 2 полями

Создайте RDD на основе коллекции

Создайте DataFrame, прочитав ранее созданный файл JSON

Создайте DataFrame, прочитав ранее созданный файл CSV

Создайте DataFrame на основе ранее созданного RDD. При этом явно определите схему (StructType)

На основе всех созданных RDD создайте временные таблицы Spark SQL

По усмотрению можно добавить дополнительные Spark SQL таблицы, используя любые источники данных

Выполните, как минимум, 5 SQL-запросов, обязательно используя в каждом, как минимум, одну из конструкций: JOIN, GROUP BY, WHERE, ORDER BY. Каждая из перечисленных конструкций должна быть использована, как минимум, 1 раз

Результирующие наборы данных сохраните в виде DataFrame

Сохраните некоторые из итоговых наборов в формате parquet и avro

При помощи стандартных средств операционной системы продемонстрируйте содержимое созданных на предыдущем шаге структур данных, представляющих собой файл parquet

Шаг 1

Установите Spark на кластер Hadoop, созданный в предыдущей практической работе, Сконфигуруйте и подготовьте к работе Spark SQL

Поменял конфигурацию docker-compose с предыдущих шагов, добавил:

spark:

image: uhopper/hadoop-spark

container\_name: spark

ports:

- 4040:4040

environment:

- CORE\_CONF\_fs\_defaultFS=hdfs://namenode:9000

- YARN\_CONF\_yarn\_resourcemanager\_hostname=resourcemanager

command: tail -f /var/log/dmesg

теперь у нас есть spark работяющий с кластером Hadoop

Шаг 2

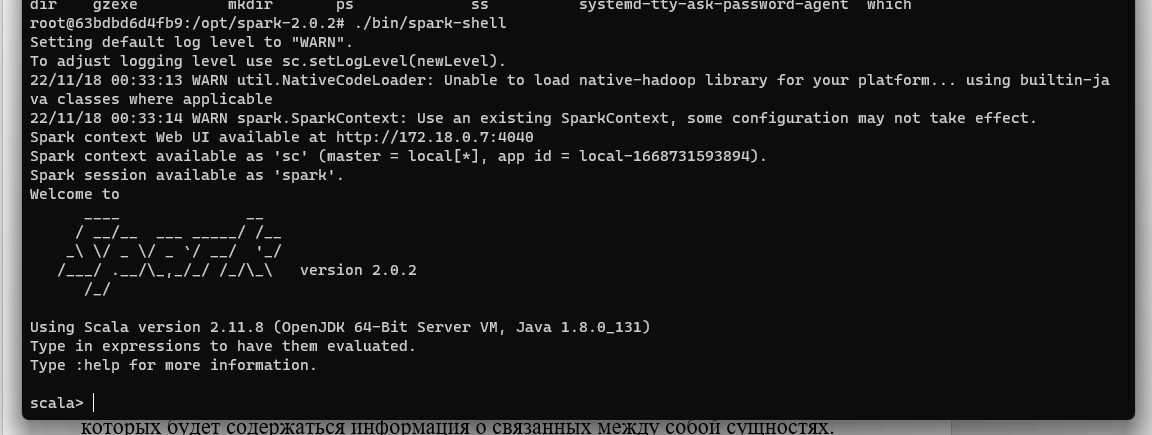
Запустите интепретатор Scala/Python и выполнить простой код (например, подсчет количества слов в тексте)

Запустим docker-compose и зайдём в контейнер со Spark

Выполним:

./bin/spark-shell

Запуститься scala интерпретатор

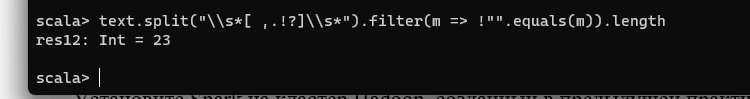


Добавим некий текст

var text = "Apache Spark is a lightning-fast unified analytics engine for big data and machine learning. It was originally developed at UC Berkeley in 2009."

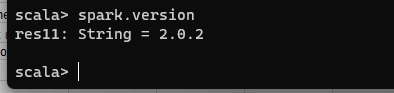
Выполним программу:

text.split("\\s\*[ ,.!?]\\s\*").filter(m => !"".equals(m)).length



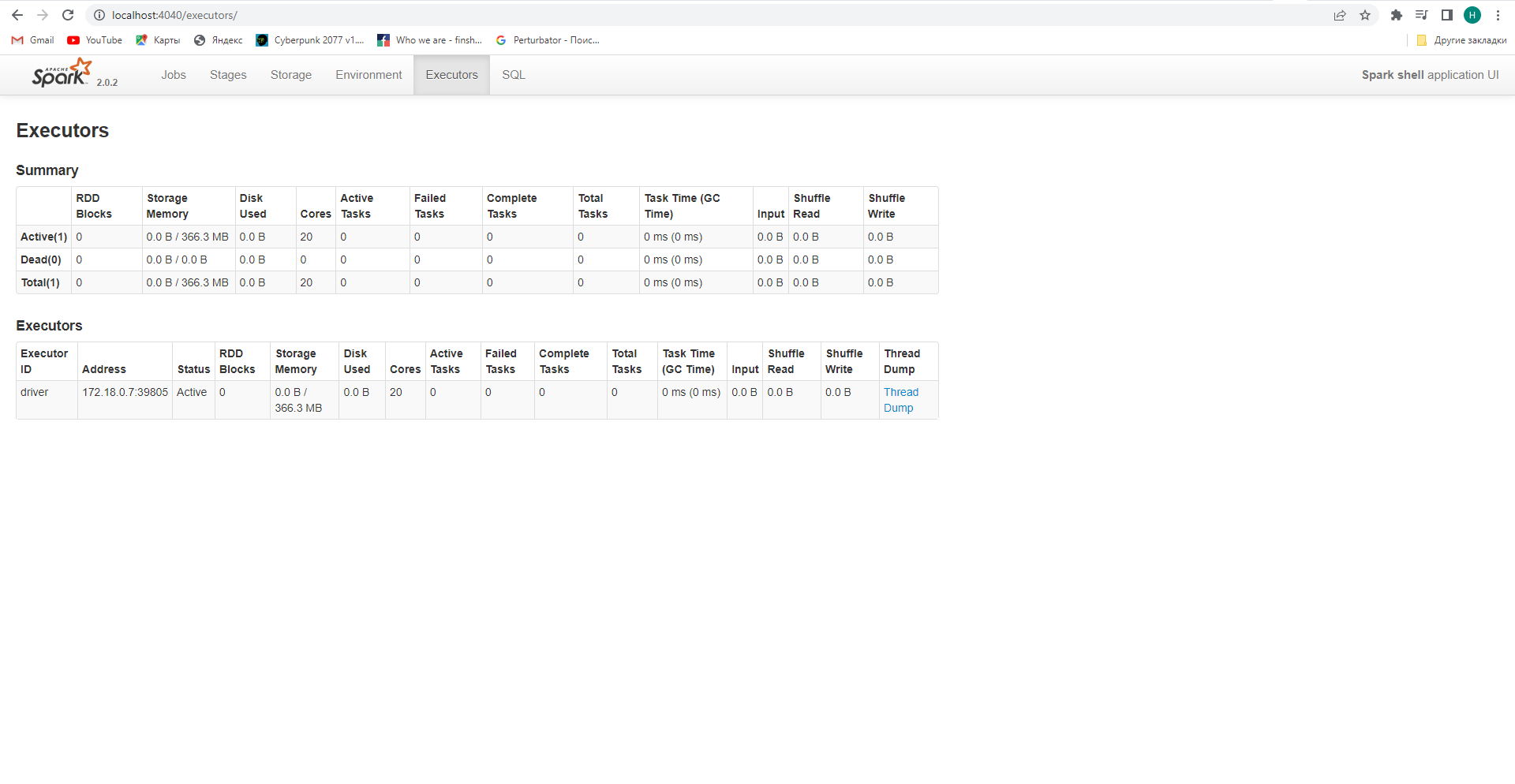
Шаг 3

При помощи встроенных средств выведите информацию о Spark в консоль



Шаг 4

Выведите web-страницу с информацией о Spark и его текущей загрузке



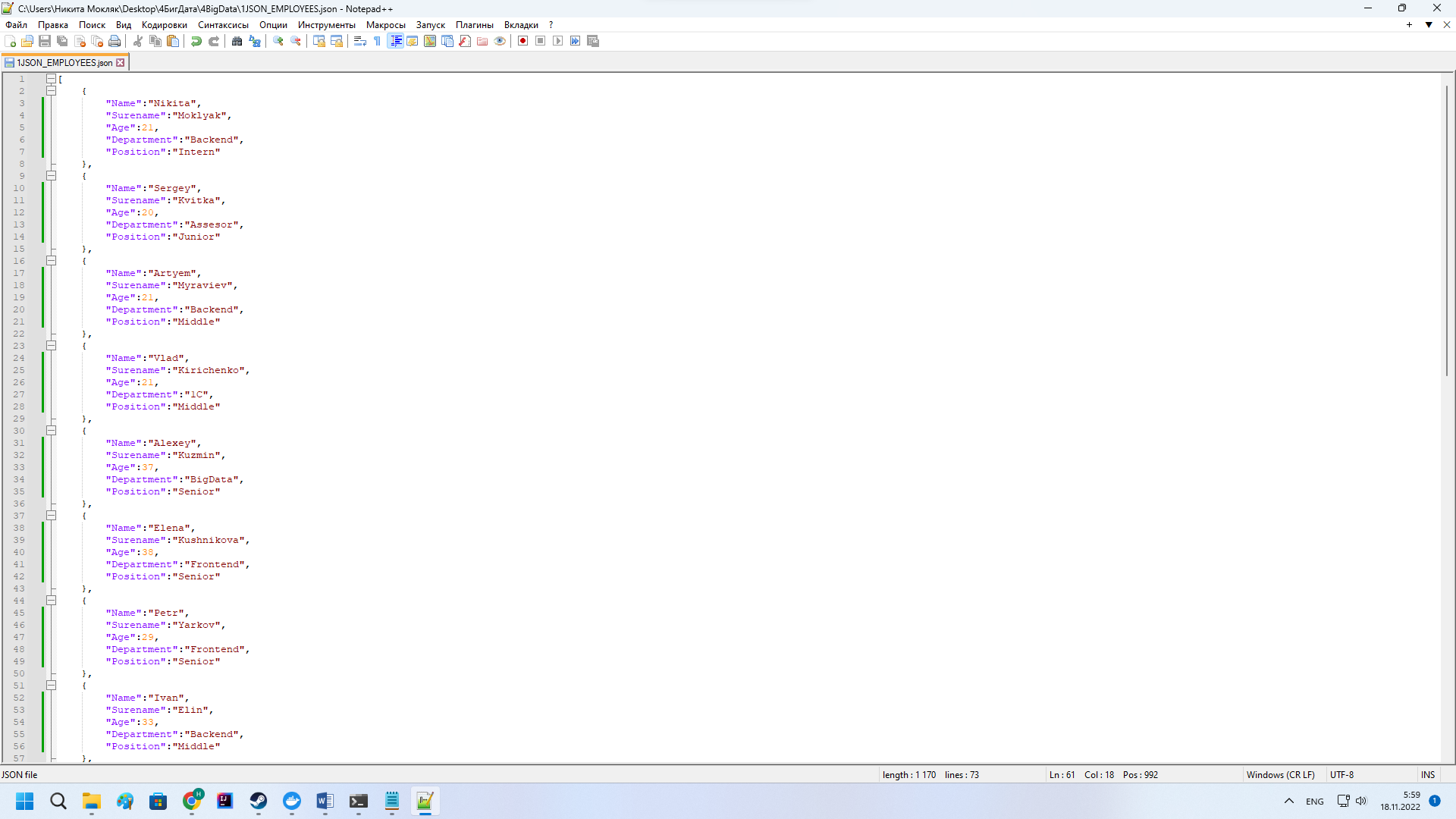
Шаг 5

Выберите предметную область и продумайте, как минимум, три таблицы, в которых будет содержаться информация о связанных между собой сущностях. Например, это может быть схема "сотрудники-отделы-должности" или "товары-категории-тары" и т.п.

сотрудники-отделы-должности

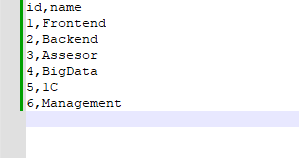
Шаг 6

Подготовьте JSON-файл, содержащий массив объектов для одной из таблиц. Каждый объект должен включать в себя порядка 3-4 полей разных типов. Всего объектов в массиве должно быть порядка 10



Шаг 7

Подготовьте CSV-файл, содержащий информацию об объектах для одной из таблиц. При этом самостоятельно выберите символы, которые будут служить разделителями полей и строк.



Разделители - запятые

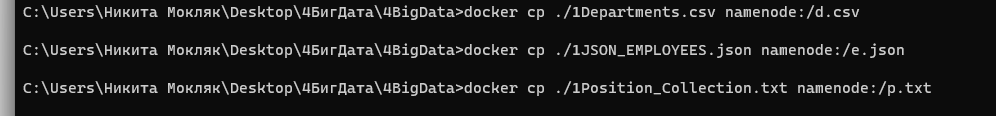
Шаг 8

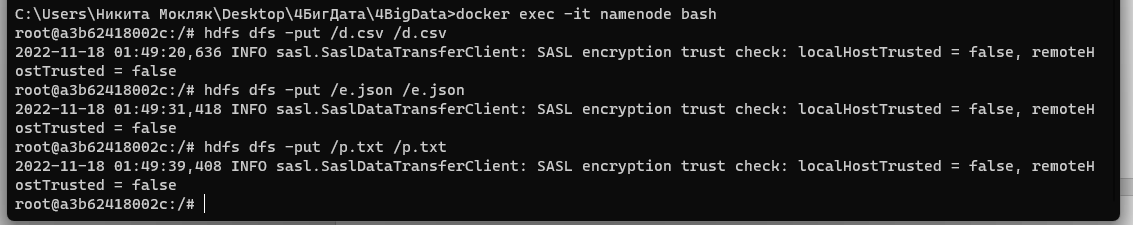
Объявите коллекцию, включающую в себя информацию об объектах для одной из таблиц. Всего должно быть более 5 объектов, с, как минимум, 2 полями



Шаг 9,10,11 инит

Копируем все файлы в hdfs





Шаг 9

Создайте RDD на основе коллекции

import org.apache.spark.sql.{Row, SparkSession}

val positionsRdd = sc.parallelize(

Seq(

Row(1, "Intern"),

Row(2, "Junior"),

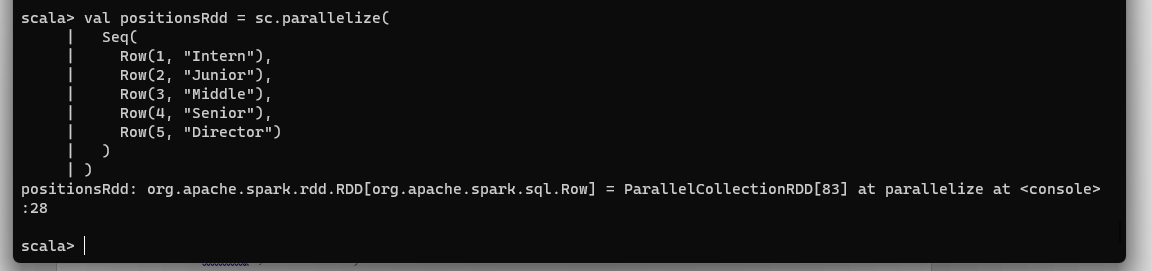
Row(3, "Middle"),

Row(4, "Senior"),

Row(5, "Director")

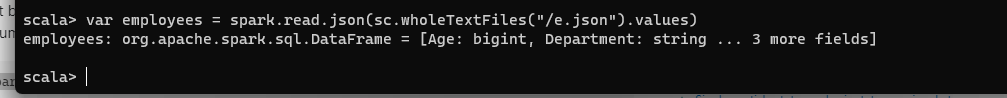
)

)



Шаг 10

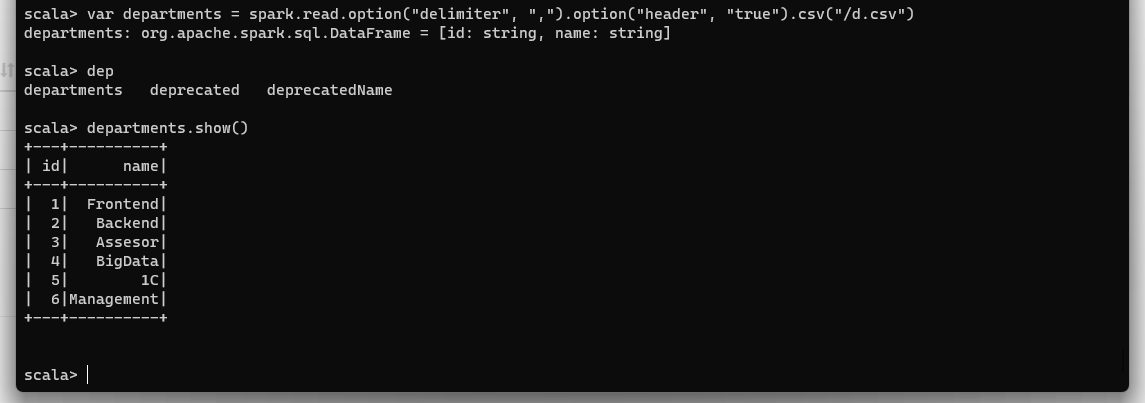
Создайте DataFrame, прочитав ранее созданный файл JSON





Шаг 11

Создайте DataFrame, прочитав ранее созданный файл CSV



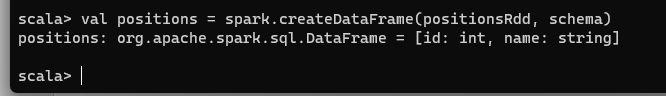
Шаг 12

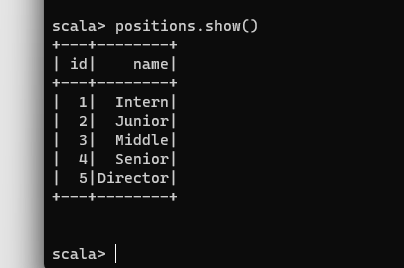
Создайте DataFrame на основе ранее созданного RDD. При этом явно определите схему (StructType)

import org.apache.spark.sql.types.{DoubleType, StringType, StructField, StructType, IntegerType}

var schema = StructType(Array(new StructField("id", IntegerType, false), new StructField("name", StringType, fals

e)))



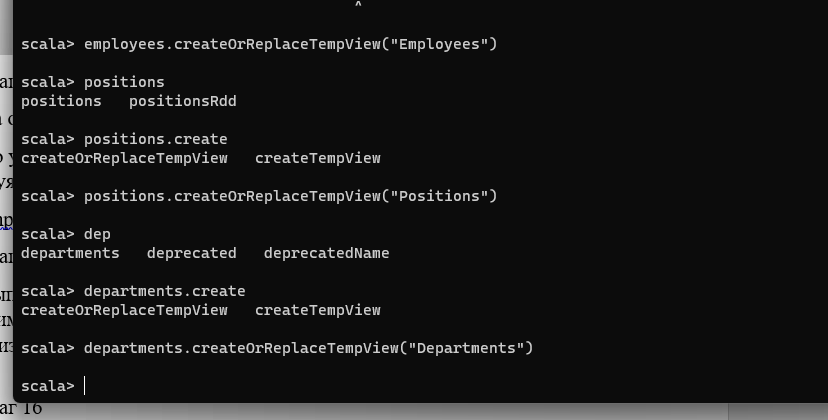


Шаг 13

На основе всех созданных DF создайте временные таблицы Spark SQL

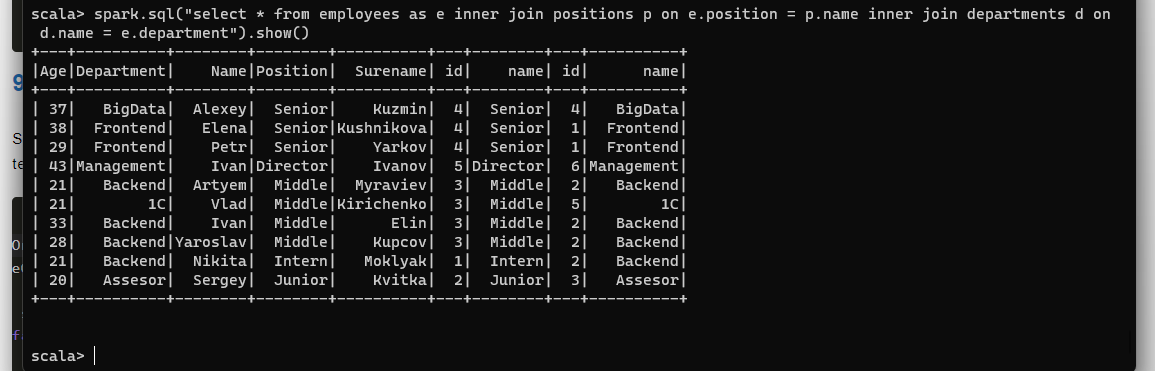
По усмотрению можно добавить дополнительные Spark SQL таблицы, используя любые источники данных

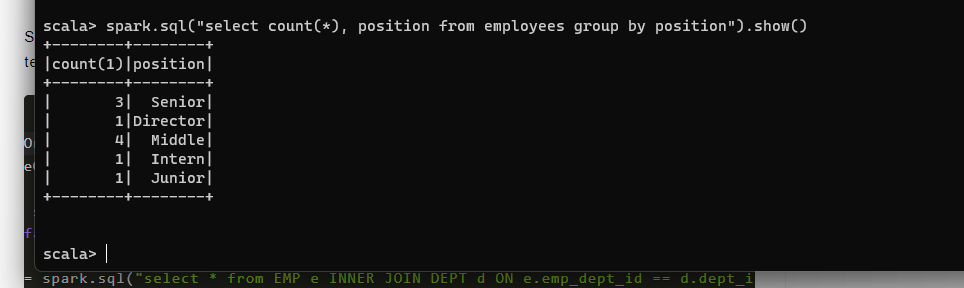
employees.createOrReplaceTempView("Employees") …

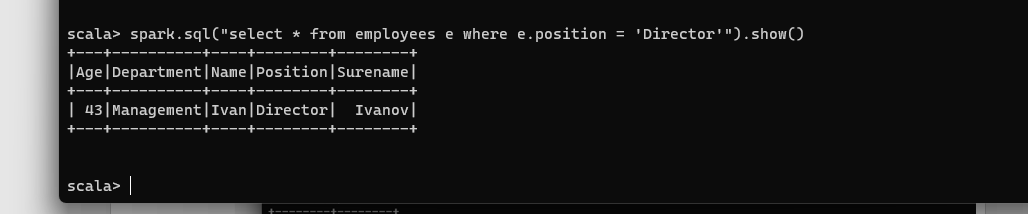


Шаг 15

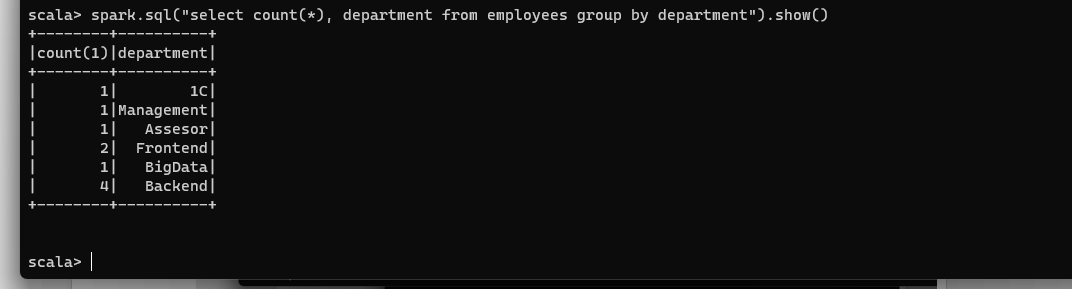
Выполните, как минимум, 5 SQL-запросов, обязательно используя в каждом, как минимум, одну из конструкций: JOIN, GROUP BY, WHERE, ORDER BY. Каждая из перечисленных конструкций должна быть использована, как минимум, 1 раз











Шаг 16

Результирующие наборы данных сохраните в виде DataFrame

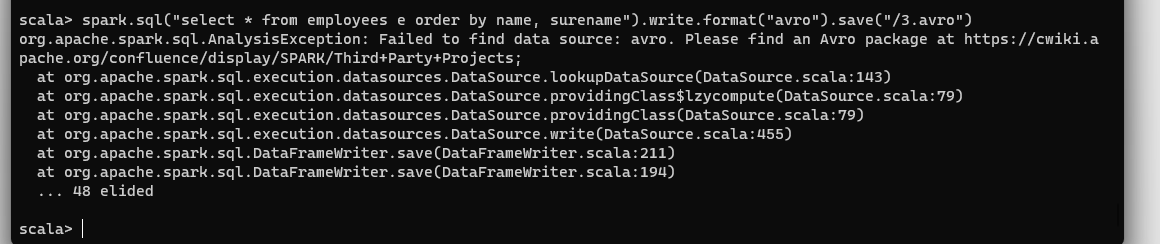
Так spark.sql() возвращает df …

Шаг 17

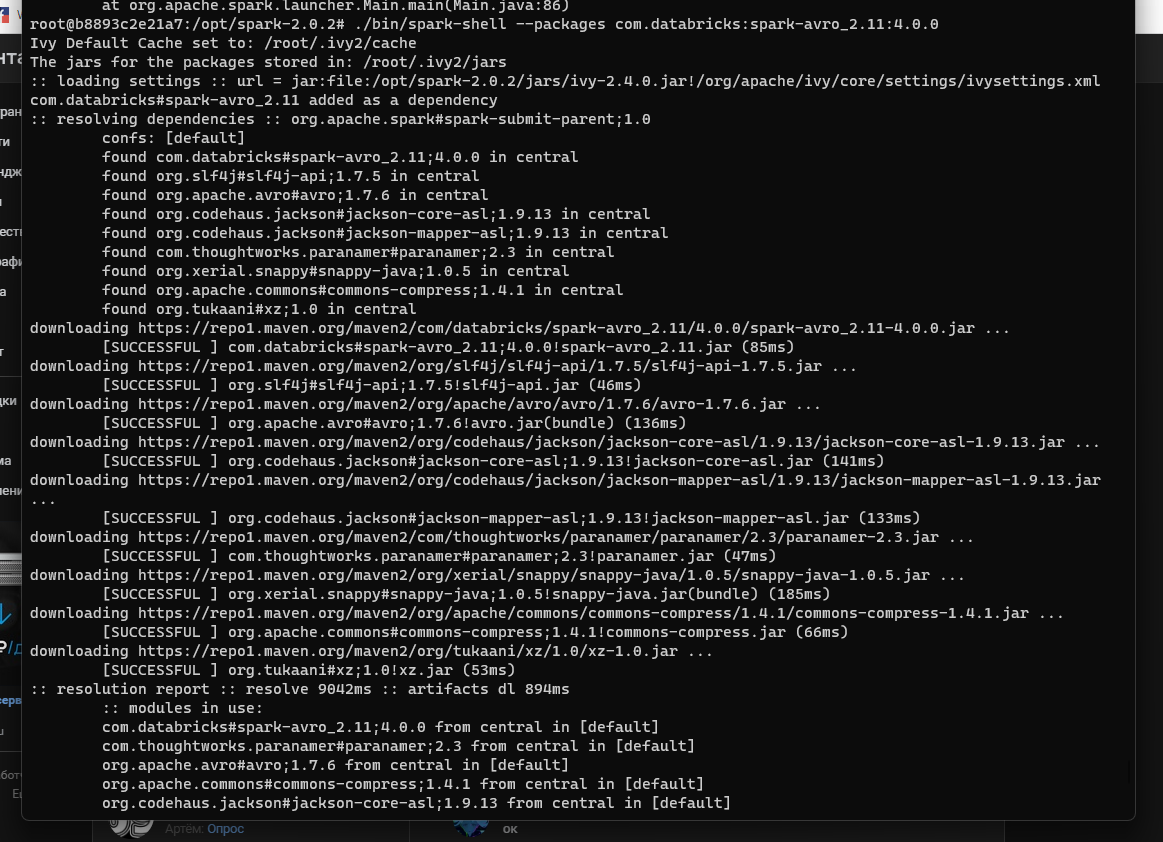
Сохраните некоторые из итоговых наборов в формате parquet и avro



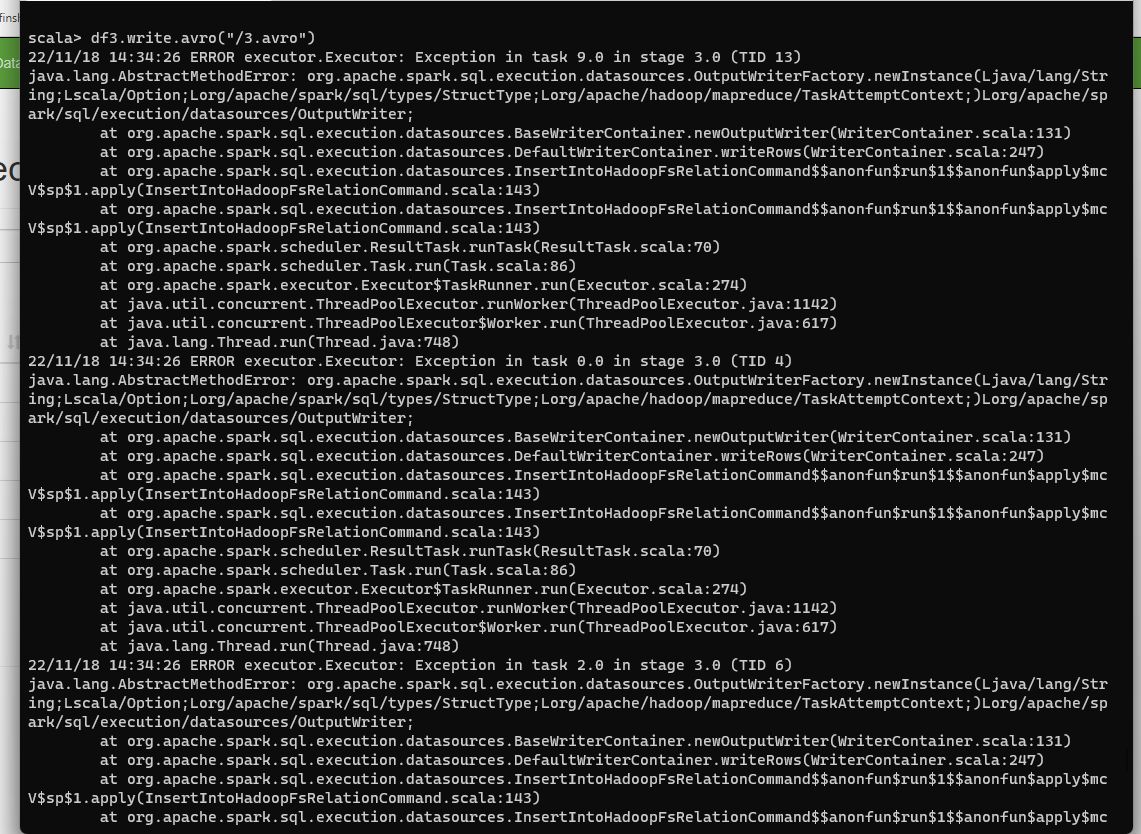
* Parquet (default)
* Avro



Не установлен ТТ



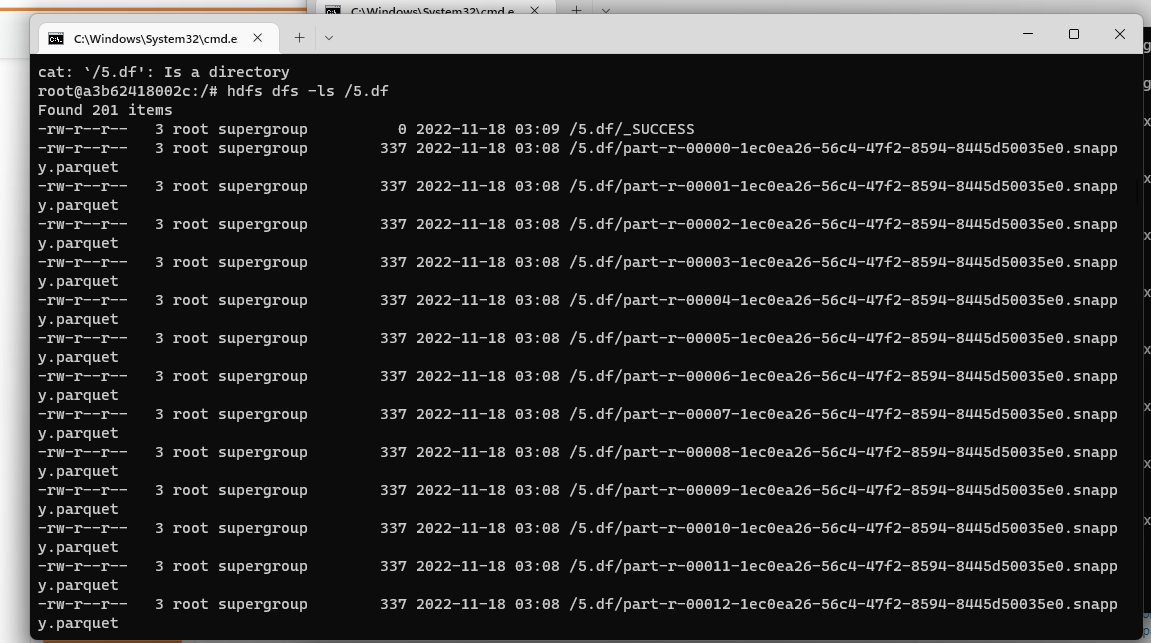
import com.databricks.spark.avro.\_

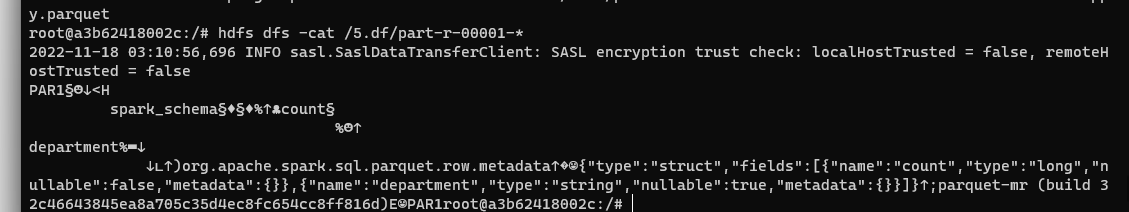


Шаг 18

При помощи стандартных средств операционной системы продемонстрируйте содержимое созданных на предыдущем шаге структур данных, представляющих собой файл parquet

parqet





Вывод

Мы научились устанавливать Spark совместно с Hadoop HDFS

Мы обучились основам работы и принципам Spark SQL. Сохранили получившиеся результаты в двух форматах: Parquet – стандартный для Hadoop и avro – внешний модуль.