МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное   
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики  
Кафедра технической кибернетики

**Отчет по лабораторной работе №2**

Дисциплина: «Большие данные»

Тема: **«Формирование отчётов в Apache Spark»**

Выполнил: Печенин В.А.

Группа: 6133-010402D

Самара 2020

**ЗАДАНИЕ**

1. Преобразовать любой файл набора данных stackoverflow в parquet формат c помощью Apache Spark.

2. Сформировать отчёт с информацией о частоте обсуждения 10 наиболее популярных языков программирования в каждом году с 2010 года по сегодняшний день. Используйте теги входящие в список языков перечисленных в википедии

https://en.wikipedia.org/wiki/List\_of\_programming\_languages.

Исходный набор данных: <https://archive.org/details/stackexchange>. В архиве с заданиями в папке data доступен пример данных: posts\_sample.xml.

Ссылки на источники:

1. https://spark.apache.org/docs/latest/sql-programming-guide.html

2. http://timepasstechies.com/spark-dataset-api-examples-tutorial-20/

3. https://jaceklaskowski.gitbooks.io/mastering-spark-sql/

4. https://en.wikipedia.org/wiki/OLAP\_cube

5. http://homepage.cs.latrobe.edu.au/zhe/ZhenHeSparkRDDAPIExamples.html

**ХОД РАБОТЫ**

Используя данные со страницы <https://en.wikipedia.org/wiki/List_of_programming_languages> был сформирован текстовый документ ***List\_of\_programming\_languages.txt***. Данный файл, а так же файл ***posts\_sample.xml*** были помещены на диск локальный диск D (рисунок 1).

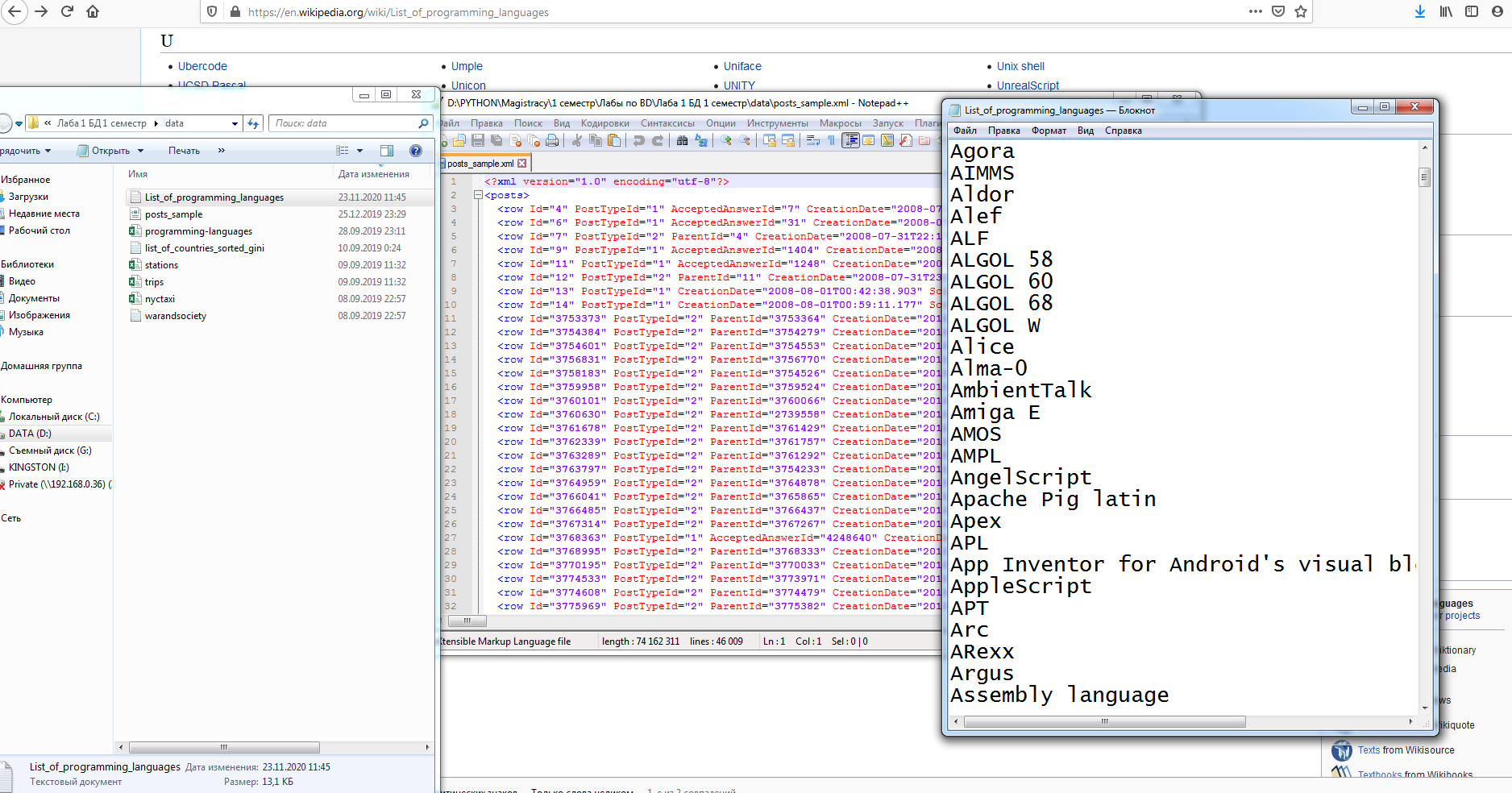


Рисунок 1 – Данные для анализа

В Intellij IDEA Community Edition был создан проект **spark-lab-2-languages** на языке Scala и с использованием системы сборки SBT (версия Scala 2.11.8). в котором будет формироваться отчёт с информацией о частоте обсуждения 10 наиболее популярных языков программирования. В файл build.sbt была дополнительно добавлена зависимость "***spark-sql***" версии 2.3.1 (рисунок 2).

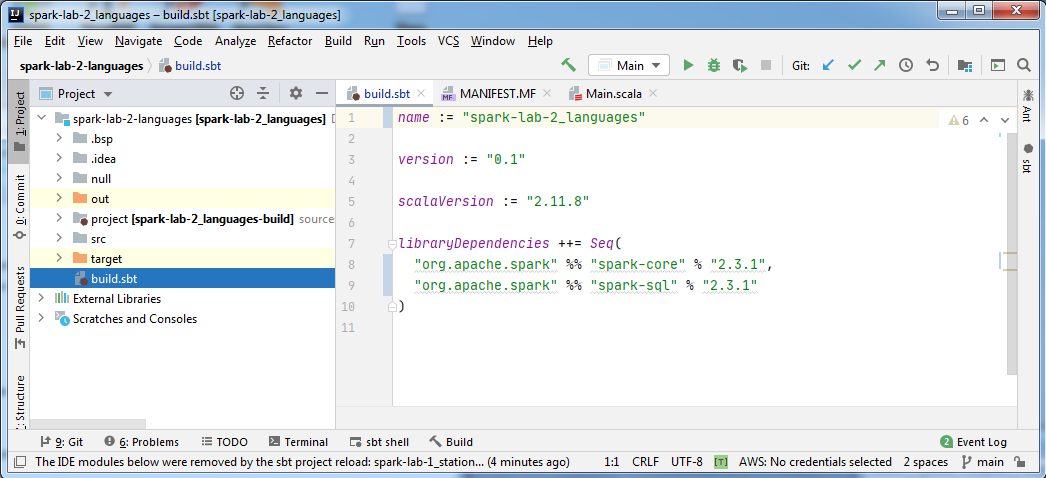


Рисунок 2 – Необходимые зависимости

В папку **null/bin** проекта поместил ***winutils.exe***. Выполнил настроки путей к локальным данным (рисунок 3).

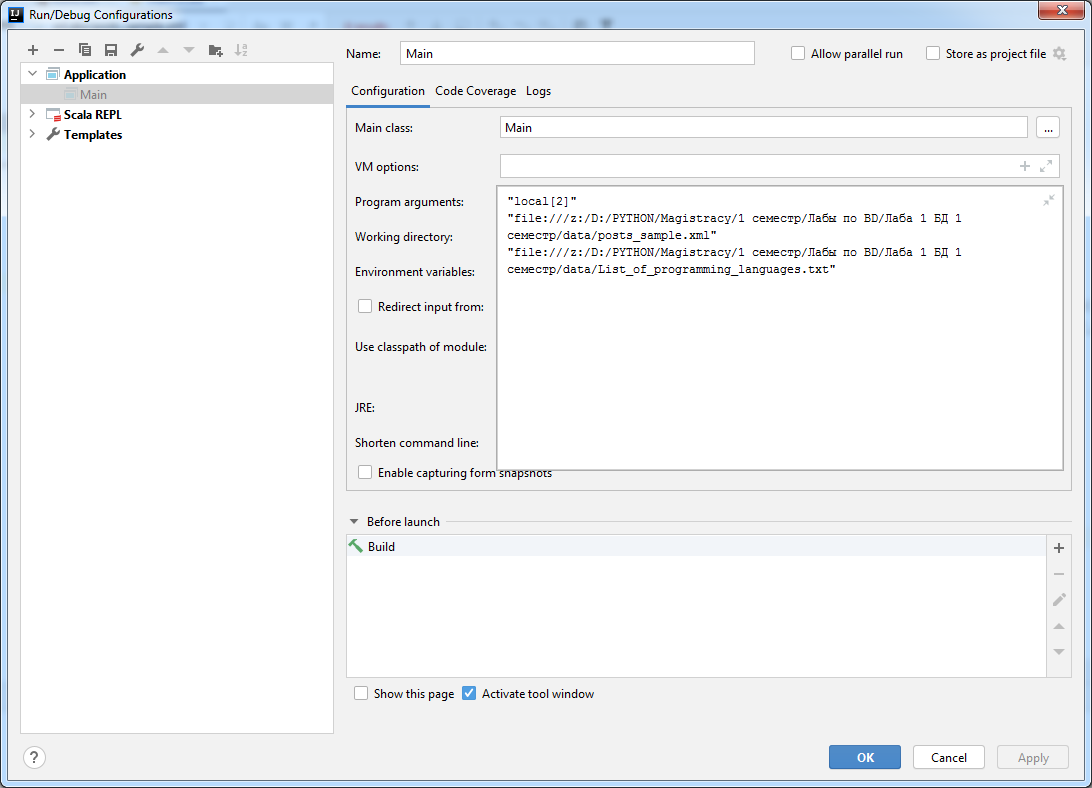


Рисунок 3 – Настройки путей

Для выполнения задания был использован Spark SQL. Spark SQL – это модуль Apache Spark, интегрирующий реляционную обработку данных и процедурный API Spark.

Был выполнен импорт необходимых библиотек **org.apache.log4j.Level**, **org.apache.log4j.Logger** и **org.apache.spark.\_**. Весь код помещен в методе ***Main*** объекта ***Main***. Для конвертации RDD в DataFrame был использован **sqlContext.implicits.\_**.

Был сформирован вектор строковых значений годов ***years***, с 2010 по 2020 г. Переменная, характеризующая количество языков (от наиболее часто упоминаемого и ниже) ***numberOfTopLanguages***. Получение данных в виде RDD в объект ***postsData*** было выполнено с использованием функции **sc.textFile**, как и было подсказано в задании, количество данных для отладки кода сначала бралась небольшая часть с помощью метода **mapPartitions(\_.take(100))**. Подсчет количества записей данных велось с помощью метода **count**. Добавление индексов к строкам и фильтрация строк без индекса из **postsData** осуществлялось с помощью метода **zipWithIndex.filter**. В объект ***languagesData*** был импортирован перечень языков из файла ***List\_of\_programming\_languages.txt***. Названия языков были опущены в нижний регистр с использованием **map{idx => idx.toLowerCase }.collect()**. Для парсинга ***xml*** строк (объект ***postsDataXml)*** был использован метод **scala.xml.XML.loadString**. На рисунке 4 приведен описанный блок вычислений.

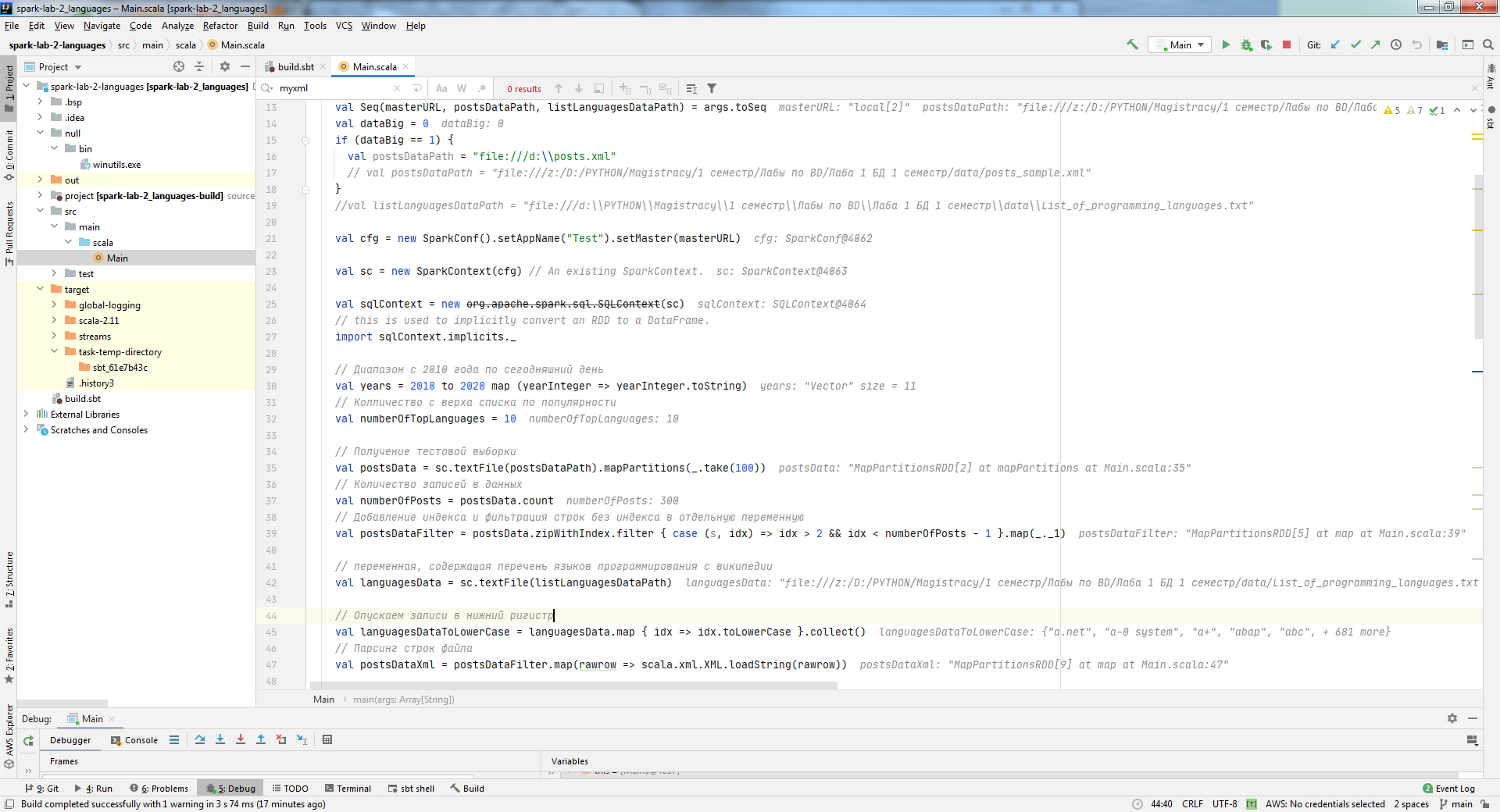


Рисунок 4 – Импорт необходимых библиотек, данных и первые операции над ними

Для извлечения из данных языки – дата пфиксации была разработана функция distributionByYears, сохраняющая эти значения в виде строк типа:

***(&lt;html&gt;&lt;css&gt;&lt;internet-explorer-7&gt;,2008-07-31T22:08:08.620)***

***(&lt;c#&gt;&lt;.net&gt;&lt;datetime&gt;,2008-07-31T23:40:59.743)***

Символами ***&gt;*** и ***&lt;*** разделяются отдельные названия языков. Для создания пар год – массив значений языкой (на год из даты) создана функция **distributionByYears**, в которой первые 4 символа из даты – получаем год, а из строки языков убираются символы ***&gt;*** и ***&lt;***. Получаются данные вида: **(2008,[Ljava.lang.String;@f559c74)**. Для разделения названий и создания отдельных пар год-язык использована конструкция **flatMap { case (year, tags) => tags.map(tag => (year, tag)) }**. Для фильтрации по данным из файла ***List\_of\_programming\_languages.txt*** используется фильтр **filter{case(year, tag) => languagesDataToLowerCase.contains(tag) }**, получая в итоге пары вида **(2008,c#)**. На рисунке 5 приведен блок кода для создания пар год-язык.

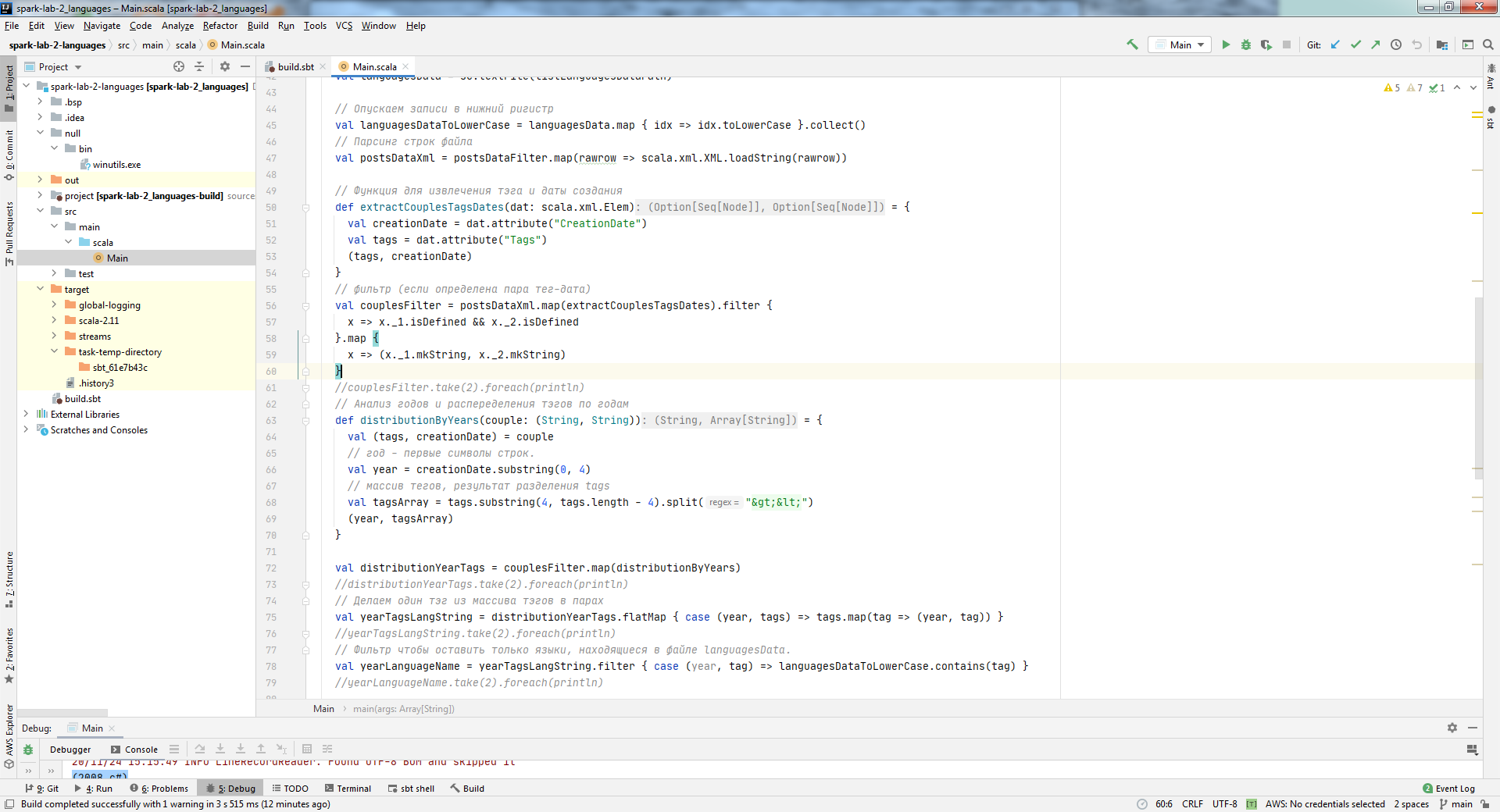


Рисунок 5 – Обработка данных для получения пар год-язык (из файла ***List\_of\_programming\_languages.txt***)

Далее с помощью операций **map**, **filter** и **reduceByKey** осуществляется создание индексированных RDD записей по годам. Затем с использованием **map**, **sortBy** и **take** производится конвертация RDD в формат вложенного списка ***topOfYearTagFrequency*** (11 основных элементов (может быть и пустыми, каждый соответствует году), и внутри каждого элемента еще список с отсортированными по частоте языками в году). С помощью операции reduce((a, b) => a.union(b)) вложенный список преобразуется в единый последовательный список (с 2010 по 2020 год). Лишние (пустые элементы) из **topOfYearTagFrequency** не входит в данный список **ListResult**.

Итогом вычислений является преобразование списка в формат DataFrame (для удобства создания отчетов) с помощью sc.parallelize(ListResult).toDF("Год", "Язык", "Количество\_упоминаний"). В **resultDataFrame** три столбца, название которых обозначены пользователем. С помощью функции **show()** в консоль выводится отчет. Учитывая, что по умолчанию выводит только первые 20 строк, максимальный размер вывода был специально задан (110 строк, 11 лет, максимально по 10 языков в топе). При необходимости (большой файл xml для обработки) результат **resultDataFrame** сохраняется в файл формата Apache Parquet. На рисунке 6 приведена заключительная часть кода.

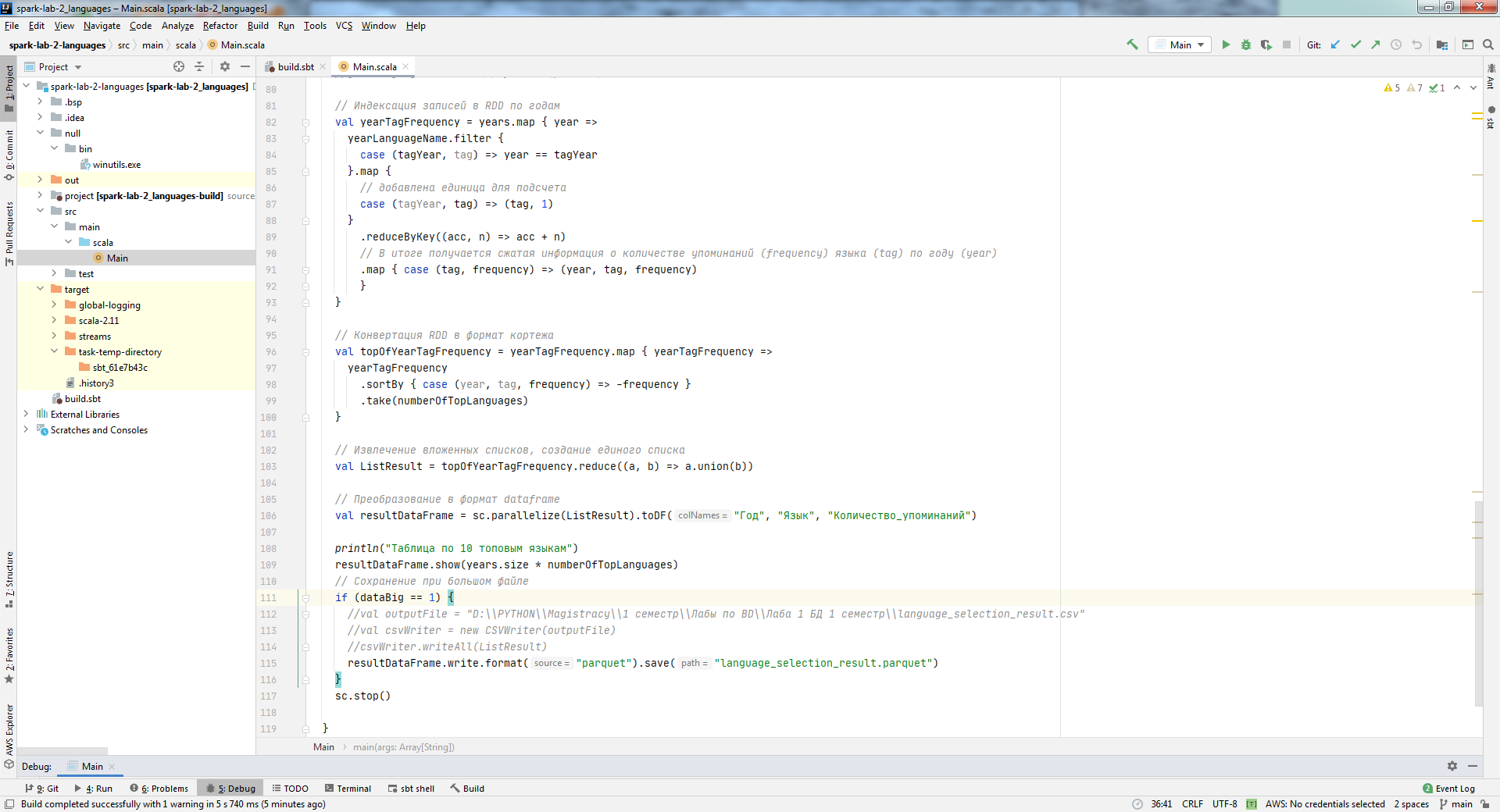


Рисунок 6 – Заключительные операции для формирования отчета

На рисунке 7 приведен результат работы для части данных из файла ***posts\_sample.xml***.

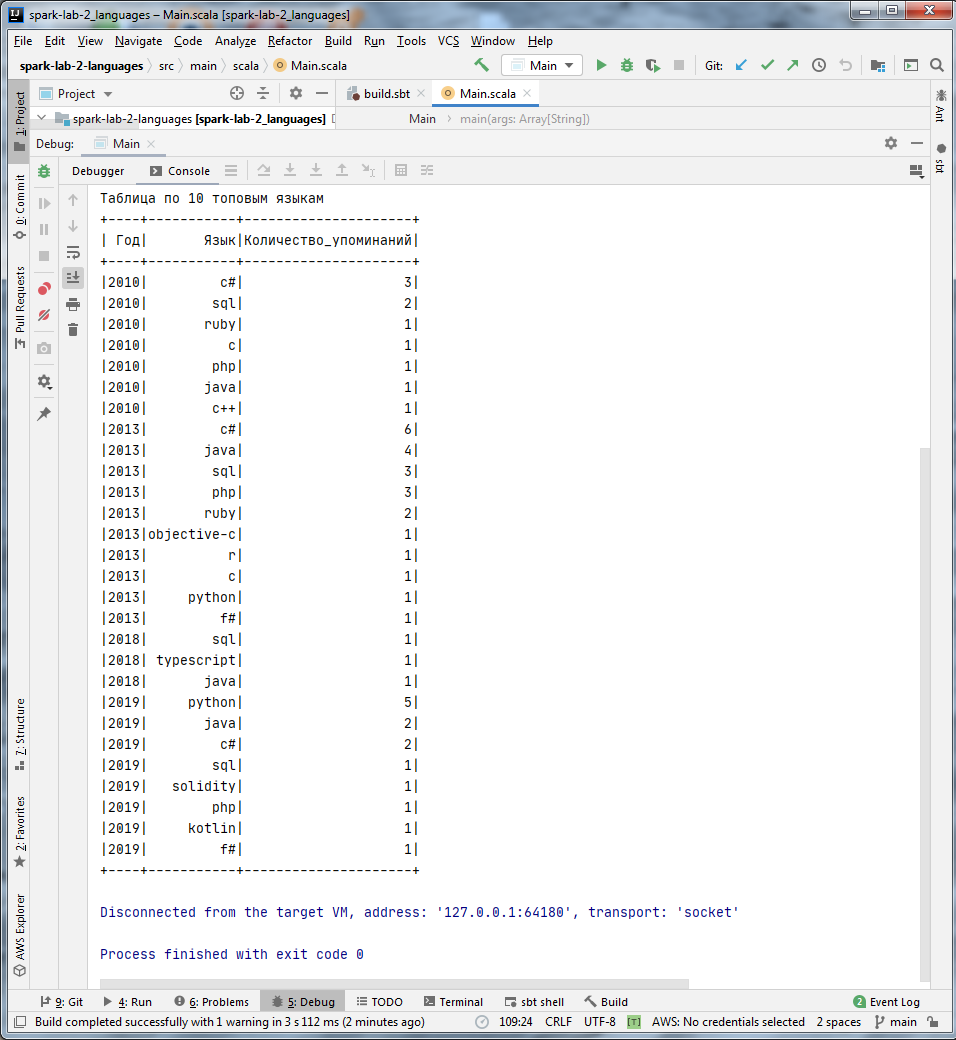


Рисунок 7 – Результирующий отчет для 100 строчек posts\_sample.xml

Скрин результатов для всего файла ***posts\_sample.xml*** приведен на рисунке 8.

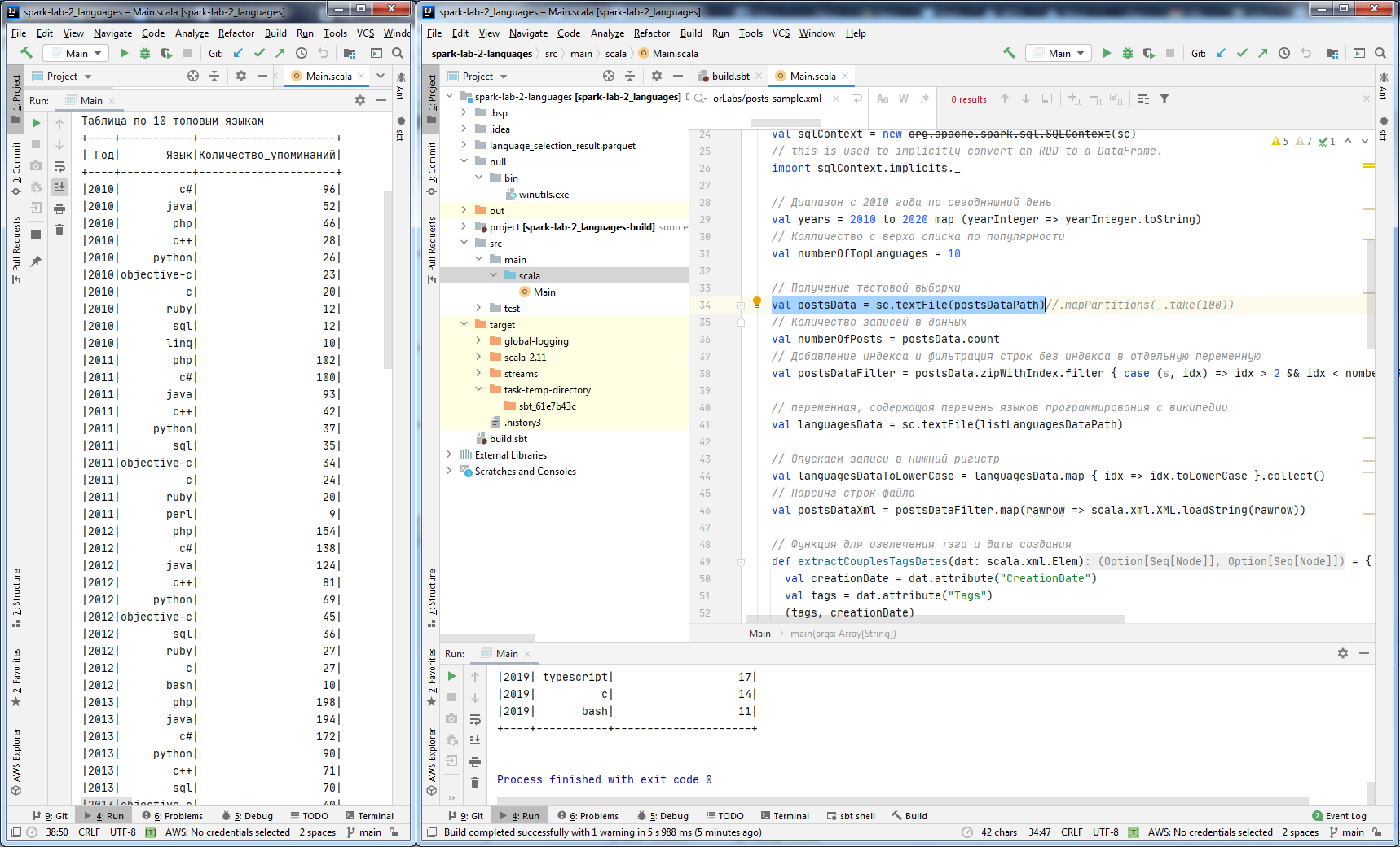


Рисунок 8 – Результирующий отчет для файла posts\_sample.xml

Был скачен файл ***Posts.xml*** c сайта <https://archive.org/download/stackexchange>. Результат отчета о частоте обсуждения 10 наиболее популярных языков программирования для каждого года с 2010 по 2020 приведен на рисунке 9.Результат был сохранен в файл ***language\_selection\_result.parquet***, полная таблица приведена в приложении А.

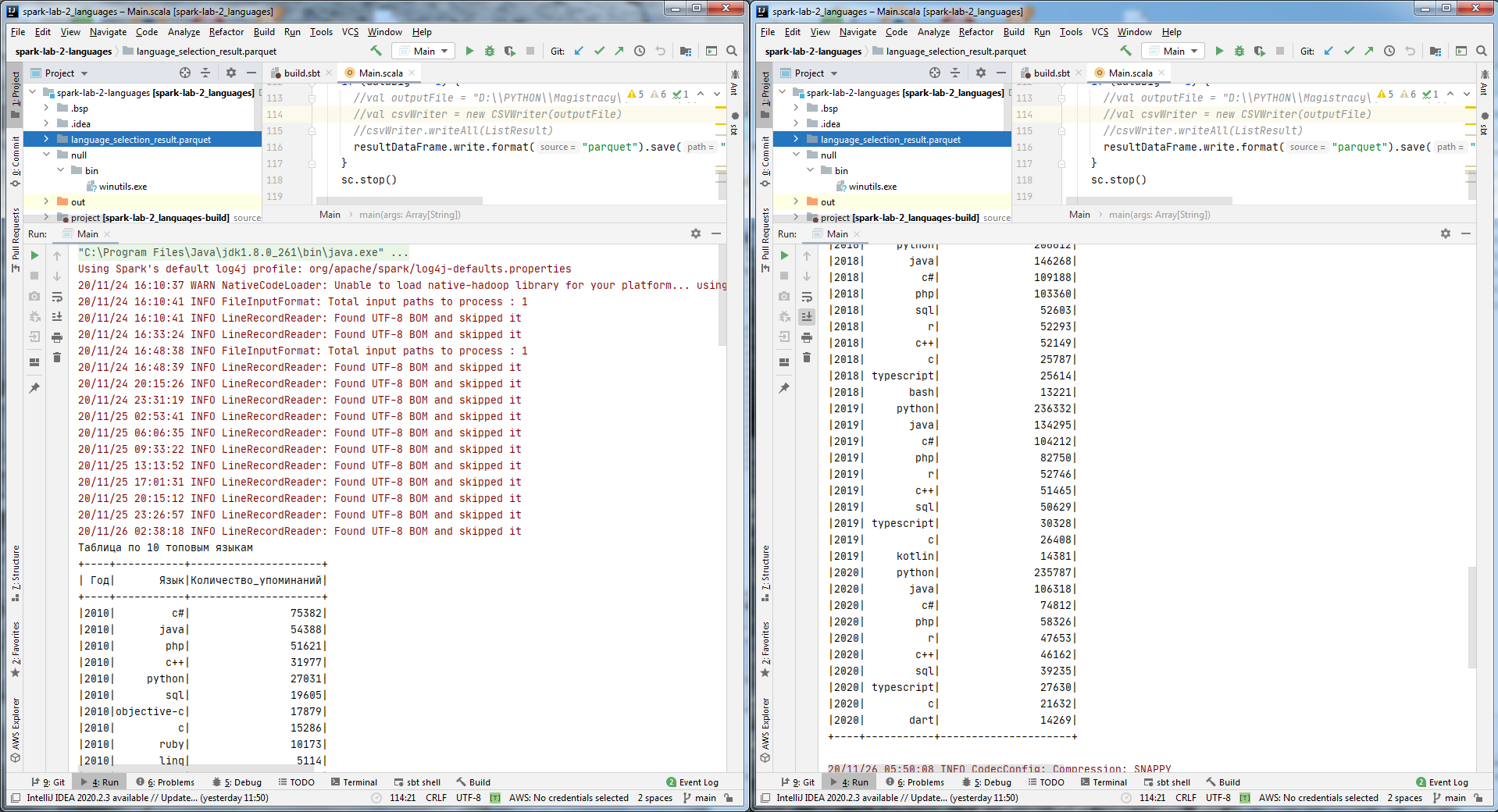
****

Рисунок 9 – Результирующий отчет для файла posts.xml

**ЗАКЛЮЧЕНИЕ**

В результате лабораторный работы были преобразованы файлы набора данных в формате .xml в parquet, сгенерирован отчет с информацией об обсуждения 10 самых популярных языков программирования. В 2010 году тройка лидеров была представлена языками **C#** (75382 раз), **Java** (54388 раз) и **PHP** (51621 раз). В 2020 году в тройку лидеров вошли **Python** (235787 раз), **Java** (106318 раз) и **C#** (74812 раз).

**ПРИЛОЖЕНИЕ А**

Результирующая таблица отчета по 10 самым обсуждаемым языкам в 2010-2020 годах

+----+-----------+---------------------+

| Год| Язык| Количество\_

упоминаний|

+----+-----------+---------------------+

|2010| c#| 75382|

|2010| java| 54388|

|2010| php| 51621|

|2010| c++| 31977|

|2010| python| 27031|

|2010| sql| 19605|

|2010|objective-c| 17879|

|2010| c| 15286|

|2010| ruby| 10173|

|2010| linq| 5114|

|2011| c#| 112857|

|2011| java| 98997|

|2011| php| 96161|

|2011| c++| 47979|

|2011| python| 42177|

|2011|objective-c| 38804|

|2011| sql| 30748|

|2011| c| 22598|

|2011| ruby| 18741|

|2011| linq| 7281|

|2012| java| 144756|

|2012| c#| 138534|

|2012| php| 131922|

|2012| python| 64260|

|2012| c++| 63143|

|2012|objective-c| 46717|

|2012| sql| 43102|

|2012| c| 31009|

|2012| ruby| 24390|

|2012| r| 12184|

|2013| java| 191046|

|2013| php| 167802|

|2013| c#| 165480|

|2013| python| 96539|

|2013| c++| 82216|

|2013| sql| 65651|

|2013|objective-c| 47553|

|2013| c| 40678|

|2013| ruby| 29304|

|2013| r| 22262|

|2014| java| 217922|

|2014| php| 179209|

|2014| c#| 163324|

|2014| python| 116578|

|2014| c++| 82872|

|2014| sql| 76210|

|2014|objective-c| 48385|

|2014| c| 41875|

|2014| r| 30928|

|2014| ruby| 30073|

|2015| java| 215678|

|2015| php| 171603|

|2015| c#| 155063|

|2015| python| 137363|

|2015| c++| 79915|

|2015| sql| 54998|

|2015| r| 40707|

|2015| c| 40103|

|2015|objective-c| 35786|

|2015| ruby| 28031|

|2016| java| 197074|

|2016| php| 161900|

|2016| python| 158646|

|2016| c#| 149226|

|2016| c++| 71591|

|2016| sql| 55919|

|2016| r| 44368|

|2016| c| 34611|

|2016| ruby| 24055|

|2016|objective-c| 23375|

|2017| python| 192255|

|2017| java| 174715|

|2017| php| 141996|

|2017| c#| 131867|

|2017| c++| 62125|

|2017| sql| 55675|

|2017| r| 51063|

|2017| c| 30648|

|2017| typescript| 21957|

|2017| ruby| 17322|

|2018| python| 206612|

|2018| java| 146268|

|2018| c#| 109188|

|2018| php| 103360|

|2018| sql| 52603|

|2018| r| 52293|

|2018| c++| 52149|

|2018| c| 25787|

|2018| typescript| 25614|

|2018| bash| 13221|

|2019| python| 236332|

|2019| java| 134295|

|2019| c#| 104212|

|2019| php| 82750|

|2019| r| 52746|

|2019| c++| 51465|

|2019| sql| 50629|

|2019| typescript| 30328|

|2019| c| 26408|

|2019| kotlin| 14381|

|2020| python| 235787|

|2020| java| 106318|

|2020| c#| 74812|

|2020| php| 58326|

|2020| r| 47653|

|2020| c++| 46162|

|2020| sql| 39235|

|2020| typescript| 27630|

|2020| c| 21632|

|2020| dart| 14269|

+----+-----------+---------------------+