Spark Assignment2 Report

Q1、

（1） 用 Spark Dataframe 计算每种氨基酸的出现频率。

Code：

```
import pyspark
pyspark.__version__
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split, col, lower, upper, regexp_replace, desc
spark = SparkSession.builder.config('spark.ui.port', 4040).appName("pyspark SQL basic example").getOrCreate()

fasta_path = "Q1_data/protein.fasta"
lines_df = spark.read.text(fasta_path)

# 过滤掉标题行，保留序列行
sequences_df = lines_df.filter(~col("value").startswith(">"))

# 将每行序列拆分为单个氨基酸字符并explode展开为多行
amino_acids_df = sequences_df.select(explode(split(col("value"), "")).alias("AminoAcid"))

# 统计氨基酸频率并排序
frequency_df = amino_acids_df.groupBy("AminoAcid").count().orderBy(desc("count"))

frequency_df.show(5)
spark.stop()
```

Result：

```
+---------+-------+
|AminoAcid|  count|
+---------+-------+
|        A|3223081|
|        L|2851645|
|        T|2795042|
|        V|2760761|
|        S|2747798|
+---------+-------+
only showing top 5 rows
```

（2）使用 PySpark 统计特定序列基序（即 “STAT”）的数量。

Code:

```python
import pyspark
pyspark.__version__
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split, col, lower, upper, regexp_replace, desc,sum,length
spark = SparkSession.builder.config('spark.ui.port', 4040).appName("pyspark SQL basic example").getOrCreate()

fasta_path = "Q1_data/protein.fasta"
lines_df = spark.read.text(fasta_path)

# 过滤掉标题行，保留序列行
sequences_df = lines_df.filter(~col("value").startswith(">"))
#定义特定motif
motif = "STAT"
motif_length = len(motif)

## 使用regexp_replace方法将所有motif替换为空
motif_removed_df = sequences_df.withColumn(
    "sequence_no_motif",
    regexp_replace(col("value"), motif, "")
)

#通过新表格中各行原序列和新序列的长度差，计算每行去掉了多少个motif
motif_count_df = motif_removed_df.withColumn(
    "motif_count",
    (length(col("value")) - length(col("sequence_no_motif"))) / motif_length
)

#计算每行去掉的motif总数
motif_count_df.agg(sum("motif_count").alias("motif_count_all")).show()
spark.stop()
```

Result:

```
+---------------+
|motif_count_all|
+---------------+
|         2052.0|
+---------------+
```

Q2、

（1）使用 Spark RDD API，根据 departuredelays.csv 文件中的 origin 列对数据进行分区，并将 origin 为 ATL 的行划分到一个分区中，而其余行随机划分到另外三个分区中。

Code：

```
sc.stop()
import pyspark
import random
pyspark.__version__
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName("PartitionByOrigin").setMaster("local[*]")
sc = SparkContext(conf=conf)

dep_path = "Q2_data/departuredelays.csv"
## 使用sc.textFile生成rdd对象，适用于后续rdd处理
dep_rdd = sc.textFile(dep_path)

##过滤掉rdd中的第一行（即表头行）
header = dep_rdd.first()
departure_rdd = dep_rdd.filter(lambda line: line != header)

##自定义partitionFunc分类方法
def partitionFunc(key):
    if key == "ATL":
        return 0
    else:
        return random.randint(1,3)

##使用第三个位置的内容为key
keyedRDD = departure_rdd.keyBy(lambda row: row.split(",")[3])

##按照自定义分类方法分出四类
partitionedRDD = keyedRDD.partitionBy(4, partitionFunc)

##投影并输出
partitionedRDD.map(lambda x: x[1]).saveAsTextFile("Q2_output")

sc.stop()
```

Result：（仅截取处理结果的部分行为示例）

第一个分区（part-00000）：origin 均为“ATL”

```
01010640,-4,517,ATL,MIA
01011925,-1,636,ATL,DFW
01011245,22,636,ATL,DFW
01011405,-3,636,ATL,DFW
01011540,-4,636,ATL,DFW
01011650,-6,636,ATL,DFW
01011425,0,517,ATL,MIA
01011805,-3,636,ATL,DFW
01010700,-2,636,ATL,DFW
01011730,33,517,ATL,MIA
```

第二、三、四个分区：（part-00001——part-00003）：origin 为其他关键词的行随机分布在这三个分区内

```
01020605,-4,602,ABE,ATL
01040605,28,602,ABE,ATL
01051245,88,602,ABE,ATL
01050605,9,602,ABE,ATL
01061215,-6,602,ABE,ATL
01060625,-3,602,ABE,ATL
01070600,0,369,ABE,DTW
01080600,0,369,ABE,DTW
01081219,54,569,ABE,ORD
01091215,43,602,ABE,ATL
```

（分区二）

```
01011245,6,602,ABE,ATL
01021245,-2,602,ABE,ATL
01030605,0,602,ABE,ATL
01061725,69,602,ABE,ATL
01061230,0,369,ABE,DTW
01071725,0,602,ABE,ATL
01081230,33,369,ABE,DTW
01090600,151,369,ABE,DTW
01091725,0,602,ABE,ATL
01090625,8,602,ABE,ATL
```

（分区三）

```
01020600,-8,369,ABE,DTW
01031245,-4,602,ABE,ATL
01041243,10,602,ABE,ATL
01071230,0,369,ABE,DTW
01070625,0,602,ABE,ATL
01071219,0,569,ABE,ORD
01080625,1,602,ABE,ATL
01080607,5,569,ABE,ORD
01091230,-4,369,ABE,DTW
01101219,0,569,ABE,ORD
```

（分区四）

Q3、

（1）使用 inner_join 函数，通过 instructor_id 将两个 Dataframe（courses.csv、instructors.csv）进行连接。

Code：

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, round
spark = SparkSession.builder.config('spark.ui.port', 4040).appName("assignment2.ipynb").getOrCreate()

##数据准备
schema1 = "id STRING,title STRING,url STRING,rating FLOAT,num_reviews INT,num_published_lectures INT,created STRING,last_update_date STRING,duration STRING,instructors_id STRING,image STRING "
schema2 = "_class STRING,id STRING,title STRING,name STRING,display_name STRING,job_title STRING,image_50x50 STRING,image_100x100 STRING,initials STRING,url STRING"
courses_df = spark.read.schema(schema1).option("header", "true").option("sep", ",").csv("Q3_data/courses.csv")
instructors_df = spark.read.schema(schema2).option("header", "true").option("sep", ",").csv("Q3_data/instructors.csv")

##用一个新的表格记录列名改变后的表格，便于进行join
changed_df = instructors_df.withColumnRenamed("id","instructors_id").withColumnRenamed("title","instructors_title").withColumnRenamed("url","instructors_url")

##inner-join合成表格
joined_df = courses_df.join(changed_df, on="instructors_id", how="inner")
joined_df.show(5)
```

Result：

```
+-------------+------+--------------------+--------------------+--------+-----------+-------------------+--------------------+--------------------+--------------------+--------------------+------+--------------------+
|instructors_id|    id|               title|                url|  rating|num_reviews|num_published_lectures|             created|    last_update_date|            duration|               image|_class|         instructor
s_title|          name|        display_name|           job_title|        image_50x50|      image_100x100|           initials|     instructors_url|
+-------------+------+--------------------+--------------------+--------+-----------+-------------------+--------------------+--------------------+--------------------+--------------------+------+--------------------+
|      9685726|567828|The Complete Pyth...|/course/complete-...|4.5927815|     452973|                155|2015-07-29T00:12:23Z|      2021-03-14|  22 total hours|https://img-c.ude...|  user|       Jose P
ortilla|          Jose|        Jose Portilla|Head of Data Scie...|https://img-c.ude...|https://img-c.ude...|                 JP| /user/joseportilla/|
|     31334738|1565838|The Complete 2023...|/course/the-compl...| 4.667258|     263152|                490|2018-02-22T12:02:33Z|      2023-01-20|65.5 total hours|https://img-c.ude...|  user|       Dr. An
gela Yu|Dr. Angela|       Dr. Angela Yu|Developer and Lea...|https://img-c.ude...|https://img-c.ude...|                 DY|/user/4b4368a3-b5...|
|      4466306|625204|The Web Developer...|/course/the-web-d...|4.6961474|     254711|                616|2015-09-28T21:32:19Z|      2023-02-12|  64 total hours|https://img-c.ude...|  user|        Colt
Steele|       Colt|         Colt Steele|Developer and Boo...|https://img-b.ude...|https://img-b.ude...|                 CS|   /user/coltsteele/|
|     13952972|756150|Angular - The Com...|/course/the-compl...|4.5926924|     180257|                472|2016-02-08T17:02:55Z|      2023-02-06|34.5 total hours|https://img-c.ude...|  user|Maximilian Sc
hwar...|Maximilian|Maximilian Schwar...|AWS certified, Pr...|https://img-b.ude...|https://img-b.ude...|                 MS|/user/maximilian-...|
|     31334738|2776760|100 Days of Code:...|/course/100-days-...|4.6952515|     177568|                676|2020-01-24T10:47:21Z|      2022-11-30|  64 total hours|https://img-c.ude...|  user|       Dr. An
gela Yu|Dr. Angela|       Dr. Angela Yu|Developer and Lea...|https://img-c.ude...|https://img-c.ude...|                 DY|/user/4b4368a3-b5...|
+-------------+------+--------------------+--------------------+--------+-----------+-------------------+--------------------+--------------------+--------------------+--------------------+------+--------------------+
only showing top 5 rows
```

（2） 使用 PySpark SQL 展示在所有与“spark”（即课程名称中包含“spark”这个单词）相关且创建时间在 2018 年 1 月 1 日 00:00:00 之后的课程中，课程评分最高的讲师的 display_name（显示名称）和 job_title（职位名称）。

Code：

```
from pyspark.sql import SparkSession
from pyspark import SparkConf
from pyspark.sql.types import StructType, StructField, StringType, IntegerType

conf = SparkConf().setAppName("Spark Read MySQL").set("spark.jars", "/data/lab/mysql-connector-j-8.4.0.jar")
spark = SparkSession.builder.config(conf=conf).getOrCreate()

##把合成后的表格注册成一个临时视图，joined_courses_instructors
joined_df.createOrReplaceTempView("joined_courses_instructors")
##对创建的临时视图使用spark sql语法
highest_rated_instructor = spark.sql("""
    SELECT
        display_name, job_title
    FROM
        joined_courses_instructors
    WHERE
        LOWER(title) LIKE '%spark%'
        AND created > '2018-01-01T00:00:00Z'
    ORDER BY
        rating DESC
    LIMIT 1
""")
##显示查询结果
highest_rated_instructor.show(truncate=False)
```

Result：

```
+------------+-------------------------------------------------+
|display_name|job_title                                        |
+------------+-------------------------------------------------+
|Lara F      |SHARING MY KNOWLEDGE WITH ENTHUSIASTIC LEARNERS!|
+------------+-------------------------------------------------+
```

（3）使用 PySpark SQL 选择所有满足以下条件的课程：(a) 课程名称中包含“interview”或“interviews”这两个单词；然后 (b) 按照课程评分（课程评分需先四舍五入保留一位小数，例如 4.67748 四舍五入为 4.7）降序

排序，同时按照创建时间降序排序（最新的课程排在前面）。

Code：

```
joined_df.createOrReplaceTempView("joined_courses_instructors")
highest_quality_instructor = spark.sql("""
    SELECT
        id,title,url,ROUND(rating, 1) AS rounded_rating, num_reviews,num_published_lectures,created, last_update_date,duration,instructors_id,image
    FROM
        joined_courses_instructors
    WHERE
        LOWER(title) LIKE '%interview%'
        OR LOWER(title) LIKE '%interviews%'
    ORDER BY
        rounded_rating DESC,
        created DESC
""")
##显示查询结果
highest_quality_instructor.show(5)
```

Result：

```
+-------+--------------------+--------------------+--------------+-----------+---------------------+--------------------+---------------+----------------+--------------+--------------------+
|     id|               title|                 url|rounded_rating|num_reviews|num_published_lectures|             created|last_update_date|        duration|instructors_id|               image|
+-------+--------------------+--------------------+--------------+-----------+---------------------+--------------------+---------------+----------------+--------------+--------------------+
|4886926|Interview Oriente...|/course/interview...|           5.0|         27|                   69|2022-09-17T17:57:14Z|     2022-10-17|16.5 total hours|      57923434|https://img-c.ude...|
|4309400|CATIA V5 FOR JOBS...|/course/automotiv...|           5.0|          5|                   26|2021-09-20T12:54:23Z|     2022-12-25| 7.5 total hours|     173220222|https://img-b.ude...|
|4829150|Réaliser des inte...|/course/realiser-...|           4.9|          9|                   32|2022-08-12T14:54:06Z|     2022-09-03|   2 total hours|      79438676|https://img-c.ude...|
|4722894|"The ""BigTech"" ...|/course/the-bigte...|           4.9|         22|                   57|2022-06-07T14:53:40Z|     2023-02-02| 5.5 total hours|      35934622|https://img-b.ude...|
|4499476|Power BI Intervie...|/course/power-bi-...|           4.9|         32|                   13|2022-01-17T11:08:03Z|     2023-02-04| 5.5 total hours|     183242794|https://img-c.ude...|
+-------+--------------------+--------------------+--------------+-----------+---------------------+--------------------+---------------+----------------+--------------+--------------------+
only showing top 5 rows
```