STA323

Big Data Analysis Software and Application (Hadoop or Spark)

Report on Assignment 1

12112627 李乐平

Question 1. Here is a retail dataset. You can find the sale information from different shops in the world. Before you start to code, you may want to observe the data first, to see if there is any missing or abnormal data points.

(1). Create a DataFrame with a schema. To make it simple, we just remove all the data records which have any missing data among all columns and the records where their quantity or UnitPrice is not positive. [1 point]

Answer:

My schema is as shown below.

```
my_schema = StructType([
    StructField("InvoiceNo", IntegerType(), True),
    StructField("StockCode", StringType(), True),
    StructField("Description", StringType(), True),
    StructField("Quantity", IntegerType(), True),
    StructField("InvoiceDate", StringType(), True),
    StructField("UnitPrice", DoubleType(), True),
    StructField("CustomerID", IntegerType(), True),
    StructField("Country", StringType(), True),
])
```

Read the .csv file.

```
spark = SparkSession.builder.config('spark.ui.port', 14040).appName("pyspark
SQL basic example").getOrCreate()

df = spark.read.option("header", "true") \
              .option("dateFormat", "M/d/yyyy H:mm") \
              .schema(my_schema) \
              .csv("./hw01/Q1_data/retail-dataset.csv")
```

Remove the missing data and the illegal data.

```
df = df.dropna()[(df["UnitPrice"] > 0) & (df["Quantity"] > 0)]
```

The result dataframe is as follows.

```
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|   InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|   536365|   85123A|WHITE HANGING HEA...|       6|12/1/2010 8:26|     2.55|     17850|United Kingdom|
|   536365|    71053| WHITE METAL LANTERN|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|
|   536365|   84406B|CREAM CUPID HEART...|       8|12/1/2010 8:26|     2.75|     17850|United Kingdom|
|   536365|   84029G|KNITTED UNION FLA...|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|
```

(2). What is the total revenue from all countries in these days? [1 point]

Answer:

Now calculate the total revenue by the summation of multiples of Quantity and UnitPrice.

```
df_with_revenue = df.withColumn("Revenue", col("Quantity") * col("UnitPrice"))
total_revenue = df_with_revenue.agg({"Revenue": "sum"}).collect()[0][0]
print(f"Total revenue of all countries: {total_revenue}")

>>> Total revenue of all countries: 8911407.904000023
```

(3). List top 5 customerIDs who spend the most in these days. [1 point]

Answer:

```
df_cust =
df_with_revenue.groupBy("CustomerID").agg(sum("Revenue").alias("Revenue"))
df_cust.orderBy("Revenue", ascending=False).limit(5).show()

>>>
```

```
+----------+------------------+
|CustomerID|           Revenue|
+----------+------------------+
|     14646|280206.01999999996|
|     18102|259657.29999999996|
|     17450|194550.78999999998|
|     16446|          168472.5|
|     14911|143825.06000000003|
+----------+------------------+
```

(4). Plot the total income/revenue chart of different countries from 2010/12/1 to 2010/12/5 (the summation includes both the start and ending days). [1 point]

Answer:

First filter out the records between Dec 1st, 2010 and Dec 5th, 2010.

```
# Convert "InvoiceDate" attribute into timestamp
df_country = df_with_revenue.withColumn("TimeStamp",
to_timestamp("InvoiceDate", "M/d/yyyy H:mm"))

df_country = df_country[(df_country["Timestamp"] >= "2010-12-01 00:00:00") &
(df_country["Timestamp"] <= "2010-12-05 23:59:59")]
```

Then group by the country and draw the plot.

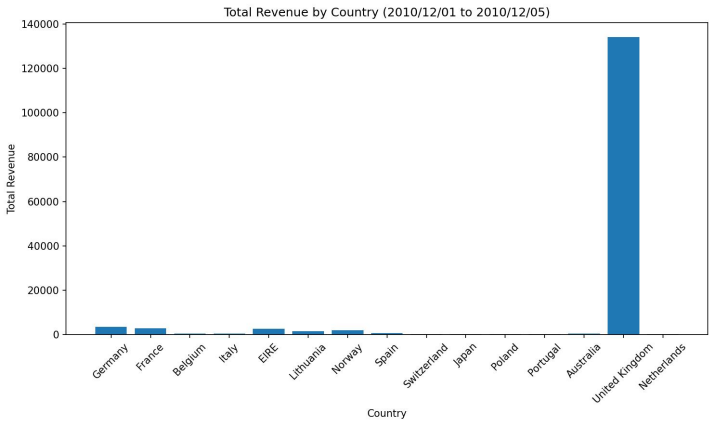
```
df_country =
df_country.groupBy("Country").agg(sum("Revenue").alias("Revenue"))

df_country_pd = df_country.toPandas()

# Draw the plot

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6), dpi = 150)
plt.bar(df_country_pd["Country"], df_country_pd["Revenue"])
plt.xlabel("Country")
plt.ylabel("Total Revenue")
plt.title("Total Revenue by Country (2010/12/01 to 2010/12/05)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Question 2. OAS is a popular protein antibody sequence dataset that contains over one billion protein sequences. Here is a small subset of the unpaired protein sequences. Please extract the columns of `sequence_alignment_aa cdr1_aa cdr2_aa cdr3_aa` in each data file (you may want to remove the first line first). Remove the lines where `cdr3_aa` length is fewer than 10 or larger than 100. Concatenate all satisfied data from different files to output one final csv file (use comma or \t to separate, only include these four columns, with titles).

(1). Use pyspark to process the data. [2 points]

Answer:

First remove the first lines of each .csv file.

```
import os
import csv

folder_path = "./hw01/Q2_data/"

csv_files = [file for file in os.listdir(folder_path) if file.endswith('.csv')
and not file.endswith("-truncated.csv")]

for file_name in csv_files:
    file_path = os.path.join(folder_path, file_name)
    output_file_path = os.path.join(folder_path,
f"{os.path.splitext(file_name)[0]}-truncated.csv")
    with open(file_path, 'r', newline='') as csvfile:
        data = list(csv.reader(csvfile))
    with open(output_file_path, 'w', newline='') as csvfile:
        csvwriter = csv.writer(csvfile)
        csvwriter.writerows(data[1:])
```

Then read the truncated .csv files and remove all the illegal records.

```
truncated_csv_files = [file for file in os.listdir(folder_path) if
file.endswith("-truncated.csv")]

dfs = [
    spark.read.option("header", "true") \
    .csv(os.path.join(folder_path, file)) \
    .select("sequence_alignment_aa", "cdr1_aa", "cdr2_aa", "cdr3_aa") \
    for file in truncated_csv_files
]

# Remove the illegal records

df2 = reduce(lambda df1, df2: df1.union(df2), dfs)
df2 = df2.filter((length("cdr3_aa") >= 10) & (length("cdr3_aa") <= 100))

df2.show()
```

```
df2.coalesce(1).write.option("header", "true").csv("./Q2_1.csv")

>>>
```

	sequence_alignment...	cdr1_aa	cdr2_aa	cdr3_aa
1	QSVLTQPPSASGTPG...	SSNIGSDT	SNN	AAWDDSLNGWV
2	QSMLTQPPSASGTPG...	NSNIGSNT	SSN	ASWDDGLDGFVI
3	QSMLTQPPSASGTPE...	NSNIGSNT	SSN	ASWDDGLDGFVI
4	GVPDRFSGSKSGTSA...			QSYDNSLSVWV
5	GVPDRFSGSTSGTSA...			QSFDNSLGGFYV

...

(2). Use linux bash script to process the data. [2 points]

Answer:

```
#!/bin/bash

folder_path="./hw01/Q2_data/"
output_folder_path="./Q2_1.csv/Q2_1.csv"
# Find CSV files
csv_files=$(find "$folder_path" -maxdepth 1 -type f -name "*.csv" ! -name
"*-truncated.csv")

# Truncate CSV files
for file_path in $csv_files; do
    output_file_path="${file_path%.*}-truncated.csv"
    tail -n +2 "$file_path" > "$output_file_path"
done

# Filter and display data
truncated_csv_files=$(find "$folder_path" -maxdepth 1 -type f -name
"*-truncated.csv")

awk -F ',' 'NR > 1 && length($47) >= 10 && length($47) <= 100 {
    print $14","$37","$41","$47
}' $truncated_csv_files > temp.csv

# Add header and write to output .csv file
header="sequence_alignment_aa,cdr1_aa,cdr2_aa,cdr3_aa"
echo "$header" > $output_folder_path
cat temp.csv >> $output_folder_path
rm temp.csv
```

The result .csv file is exactly the same as the one generated in Q2.(1).

	sequence_alignment...	cdr1_aa	cdr2_aa	cdr3_aa
1	QSVLTQPPSASGTPG...	SSNIGSDT	SNN	AAWDDSLNGWV
2	QSMLTQPPSASGTPG...	NSNIGSNT	SSN	ASWDDGLDGFVI
3	QSMLTQPPSASGTPE...	NSNIGSNT	SSN	ASWDDGLDGFVI
4	GVPDRFSGSKSGTSA...			QSYDNSLSVWV
5	GVPDRFSGSTSGTSA...			QSFDNSLGGFYV

(3). For large files like SRR12326775_1_Light_Bulk.csv , we sometimes need to split it into chunks and process each chunk. Use split shell command to equally split this file into 8 chunks, use for-loop in the shell script to parallel process each chunk. Then concatenate all chunk outputs into. In this small task, you only need to process the SRR12326775_1_Light_Bulk.csv file. [2 points]

Answer:

```
#!/bin/bash

input_file="./hw01/Q2_data/SRR12326775_1_Light_Bulk.csv"
output_folder="./Q2_3_chunks/"
final_output="./Q2_3.csv"
```

```
# Create output folder if it doesn't exist
mkdir -p "$output_folder"

# Split the input .csv file into 8 chunks
split -n l/8 -d "$input_file" "$output_folder"

# Process each chunk in parallel
for chunk_file in "$output_folder"*; do
    # Filter records based on the criteria for the 47th column
    awk -F ',' 'length($47) >= 10 && length($47) <= 100 {
        print $14","$37","$41","$47
    }' "$chunk_file" > "${chunk_file}.processed" &
done
wait

echo "sequence_alignment_aa,cdr1_aa,cdr2_aa,cdr3_aa" > "$final_output"
cat "${output_folder}"*.processed >> "$final_output"

# Remove temporary processed chunk files
rm -rf "${output_folder}"
```

The first several lines of the resulting .csv file is as shown below.

	sequence_alignment...	cdr1_aa	cdr2_aa	cdr3_aa
1	EIVMTQSPATLSVSPG...	QSVSSN	GTS	HQYNSWPPGT
2	DIQMTQSPSSLSASV...	QSISSY	AAS	QQSYSTHPYT
3	EIVMTQSPATLSVSPG...	QSVSSN	GAS	QQYNNWPPWT
4	EIVLAQSPATLSLSPGE...	QSVSSY	DAS	QQRNNWPPYT
5	DIVLTQSPGTLSLSPG...	HSINRRF	GTS	QQYDTSQGYP