STA323 Assignment1 report

SID: 12110821
Name: ZHANG Chi

Solution for Q1

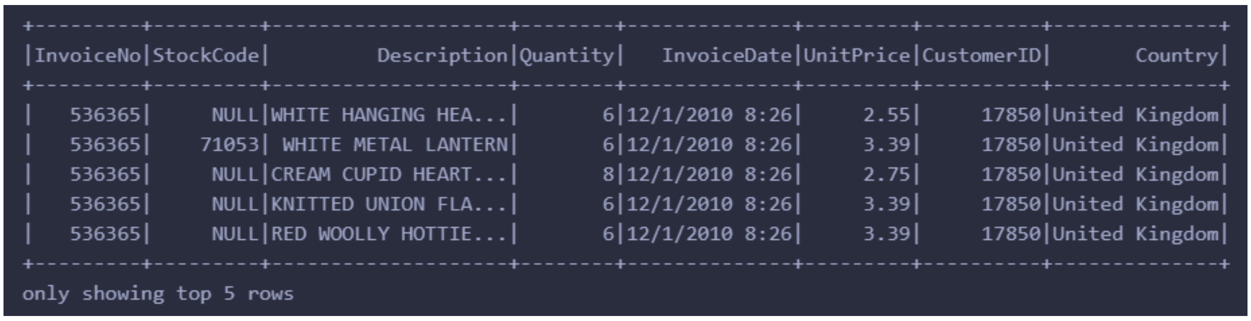
(1)

There are 8 coliuns in the dataset, and the missing value is not allowed when the data is read in. There are total 541909 rows.

The schema of the dataset is defined as follows:

```
schema = StructType([
    StructField("InvoiceNo", IntegerType(), False),     # 第三个：是否允许有空值
    StructField("StockCode", IntegerType(), False),
    StructField("Description", StringType(), False),
    StructField("Quantity", IntegerType(), False),
    StructField("InvoiceDate", StringType(), False),
    StructField("UnitPrice", FloatType(), False),
    StructField("CustomerID", IntegerType(), False),
    StructField("Country", StringType(), False)
])
df = spark.read.csv("./data/Q1_data/retail-dataset.csv", header=True,
schema=schema)
```

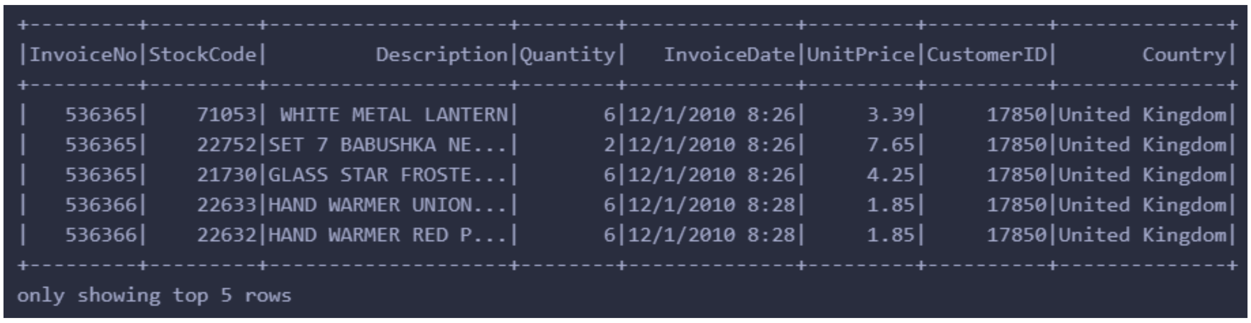
The first 5 rows of the dataset are shown below:



```
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|   InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|   536365|     NULL|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|     NULL|CREAM CUPID HEART...|       8|12/1/2010 8:26|     2.75|     17850|United Kingdom|
|   536365|     NULL|KNITTED UNION FLA...|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|
|   536365|     NULL|RED WOOLLY HOTTIE...|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
only showing top 5 rows
```

After using `filter` to drop the records where their `quantity` or `UnitPrice` is not positive, 363087 rows are left.

```
df_clean = df.dropna().filter((col("Quantity") > 0) & (col("UnitPrice")> 0))
```

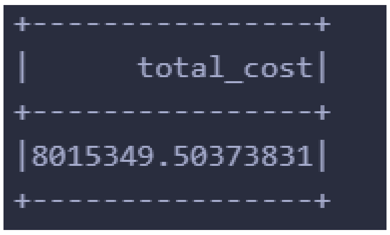


```
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|   InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|   536365|    71053| WHITE METAL LANTERN|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|
|   536365|    22752|SET 7 BABUSHKA NE...|       2|12/1/2010 8:26|     7.65|     17850|United Kingdom|
|   536365|    21730|GLASS STAR FROSTE...|       6|12/1/2010 8:26|     4.25|     17850|United Kingdom|
|   536366|    22633|HAND WARMER UNION...|       6|12/1/2010 8:28|     1.85|     17850|United Kingdom|
|   536366|    22632|HAND WARMER RED P...|       6|12/1/2010 8:28|     1.85|     17850|United Kingdom|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
only showing top 5 rows
```

(2)

The revenue is calculated by multiplying the `Quantity` and `UnitPrice` for each record, and then summing up the results.

```
df_clean.select(expr("sum(UnitPrice * Quantity) as total_cost")).show()
```



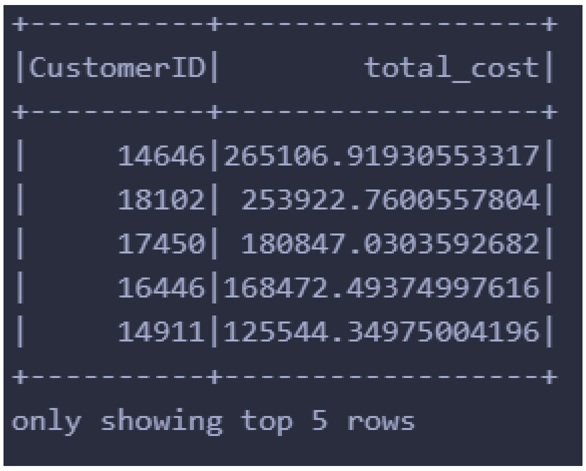
```
+----------------+
|      total_cost|
+----------------+
|8015349.50373831|
+----------------+
```

(3)

Before get top 5 customers that spend most, we need to calculate the total cost for each customer. Then we can sort the total cost in descending order.

Here I use `groupby` and agg to calculate the total cost for each customer, and then use `orderBy` to sort the result.

```
df_clean.groupBy("CustomerID").agg(expr("sum(UnitPrice * Quantity) as total_cost")).select(col("CustomerID"),col("total_cost")).orderBy("total_cost",ascending = False).show(5)
```



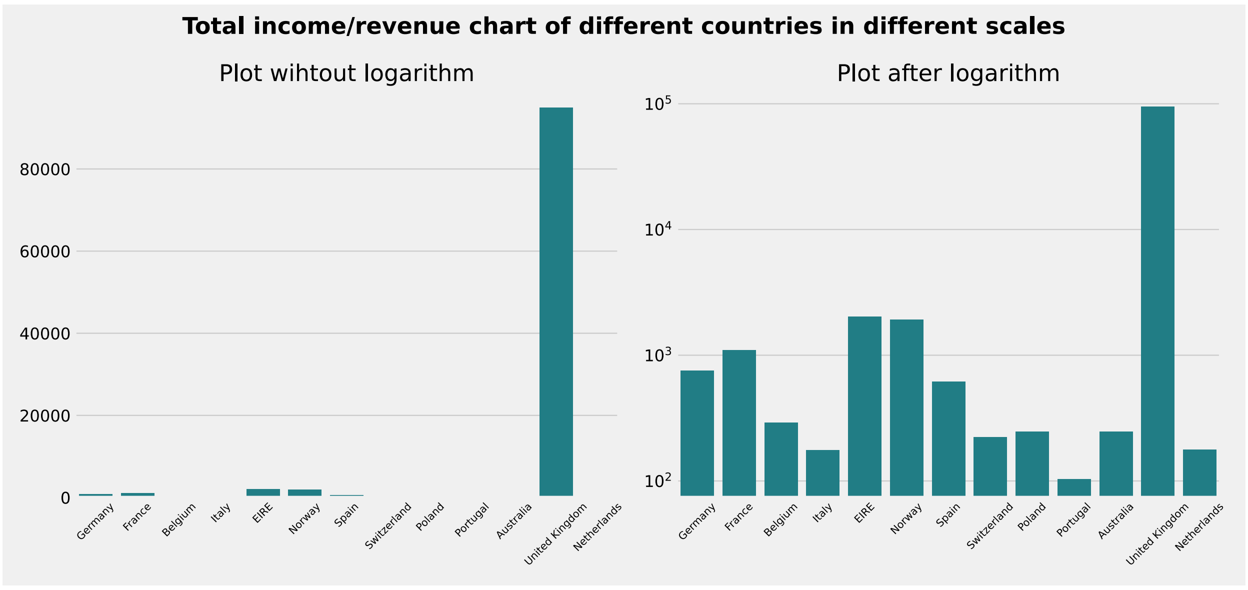
```
+----------+-----------------+
|CustomerID|       total_cost|
+----------+-----------------+
|     14646|265106.91930553317|
|     18102| 253922.7600557804|
|     17450| 180847.0303592682|
|     16446|168472.49374997616|
|     14911|125544.34975004196|
+----------+-----------------+
only showing top 5 rows
```

(4)

I use `to_timestamp` to convert the `InvoiceDate` to date type. After selecting (by `where`) the data during designated days, I use `groupBy` and `agg` to calculate the total cost for each day.

```
# set the timeParserPolicy to LEGACY
spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")
df_4 = df_clean.withColumn("InvoiceDate", to_timestamp(col("InvoiceDate"),
"MM/dd/yyyy HH:mm"))
df_4.where("InvoiceDate between '2010-12-01' and '2010-12-05'
").groupBy("Country").agg(expr("sum(UnitPrice * Quantity) as total_cost
")).select(col("Country"),col("total_cost")).collect()
```

The bar plot drawed by `seaborn` is shown below. To make it clear, I also draw the plot in `log` scale.



Solution for Q2

(1)

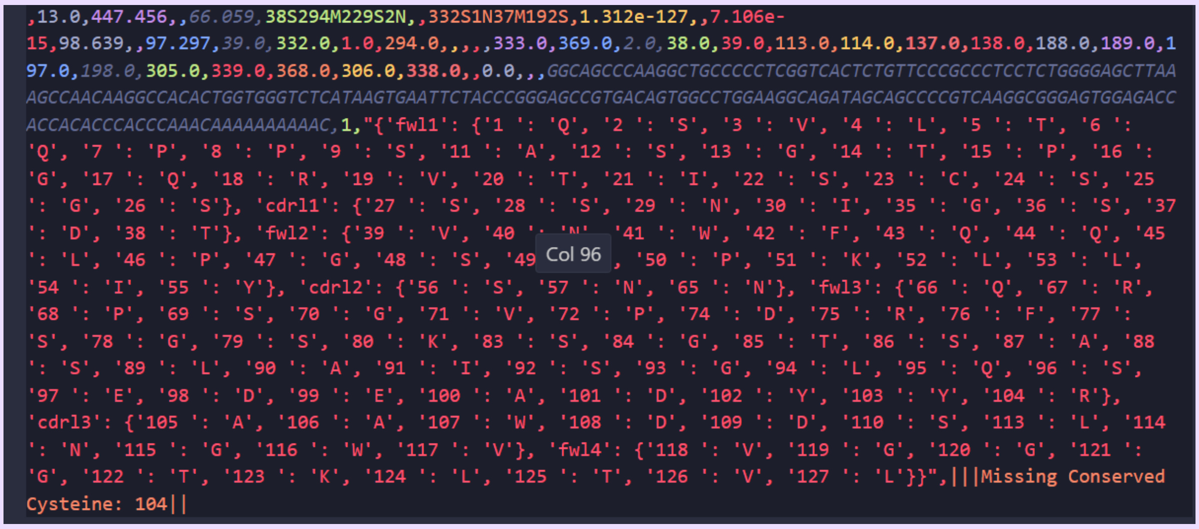
I use `os.listdir` to get the list of all CSV files in the directory. Then I read the first CSV file to initialize the DataFrame, and use `union` to merge the rest of the CSV files. Observing that the first row of each CSV file features the double quote as it beginning, to remove it, the `comment` option is used while loading `.csv` files.

```
import os
csv_list = [p for p in os.listdir(path = "data/Q2_data") if p.endswith(".csv")]

# 读取第一个 CSV 文件来初始化 DataFrame
df = 
spark.read.format("csv").option("header","True").option("comment","\"").load(f"
data/Q2_data/{csv_list[0]}")

# 从第二个文件开始，读取每一个 CSV 文件并进行合并
for i in csv_list[1:]:
    temp_df = 
spark.read.format("csv").option("header","True").option("comment","\"").load(f"
data/Q2_data/{i}")
    df = df.union(temp_df)
```

Before using `spark.read` to load files and remove first lines by `comment` , I tried to use `spark.read.text` and extract the lines by `df.tail()` (except first line) to `createDataFrame` . However, I can't find a easy way to split the each row to make the the number of result correpond to the number of columns as some values in one row is wrapped by `"{}"` with many commas(`,`) inside. The graphy is shown below. I can't split the row by a single comma.



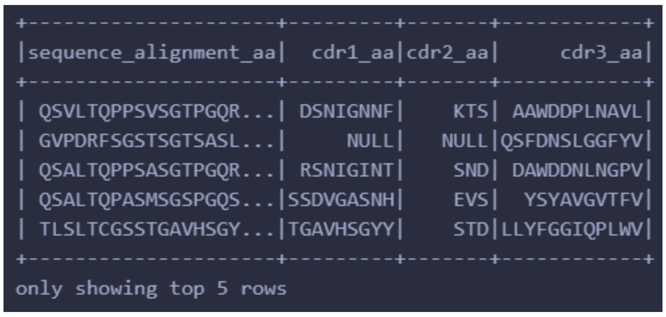
```
,13.0,447.456,,66.059,38S294M229S2N,,332S1N37M192S,1.312e-127,,7.106e-
15,98.639,,97.297,39.0,332.0,1.0,294.0,,,,,333.0,369.0,2.0,38.0,39.0,113.0,114.0,137.0,138.0,188.0,189.0,1
97.0,198.0,305.0,339.0,368.0,306.0,338.0,,0.0,,,GGCAGCCCAAGGCTGCCCCCTCGGTCACTCTGTTCCCGCCCTCCTCTGGGGAGCTTAA
AGCCAACAAGGCCACACTGGTGGGTCTCATAAGTGAATTCTACCCGGGAGCCGTGACAGTGGCCTGGAAGGCAGATAGCAGCCCCGTCAAGGCGGGAGTGGAGACC
ACCACACCCACCCAAACAAAAAAAAAAC,1,"{'fwl1': {'1 ': 'Q', '2 ': 'S', '3 ': 'V', '4 ': 'L', '5 ': 'T', '6 ':
'Q', '7 ': 'P', '8 ': 'P', '9 ': 'S', '11 ': 'A', '12 ': 'S', '13 ': 'G', '14 ': 'T', '15 ': 'P', '16 ':
'G', '17 ': 'Q', '18 ': 'R', '19 ': 'V', '20 ': 'T', '21 ': 'I', '22 ': 'S', '23 ': 'C', '24 ': 'S', '25
': 'G', '26 ': 'S'}, 'cdrl1': {'27 ': 'S', '28 ': 'S', '29 ': 'N', '30 ': 'I', '35 ': 'G', '36 ': 'S', '37
': 'D', '38 ': 'T'}, 'fwl2': {'39 ': 'V', '40 ': 'N', '41 ': 'W', '42 ': 'F', '43 ': 'Q', '44 ': 'Q', '45
': 'L', '46 ': 'P', '47 ': 'G', '48 ': 'S', '49 Col 96, '50 ': 'P', '51 ': 'K', '52 ': 'L', '53 ': 'L',
'54 ': 'I', '55 ': 'Y'}, 'cdrl2': {'56 ': 'S', '57 ': 'N', '65 ': 'N'}, 'fwl3': {'66 ': 'Q', '67 ': 'R',
'68 ': 'P', '69 ': 'S', '70 ': 'G', '71 ': 'V', '72 ': 'P', '74 ': 'D', '75 ': 'R', '76 ': 'F', '77 ':
'S', '78 ': 'G', '79 ': 'S', '80 ': 'K', '83 ': 'S', '84 ': 'G', '85 ': 'T', '86 ': 'S', '87 ': 'A', '88
': 'S', '89 ': 'L', '90 ': 'A', '91 ': 'I', '92 ': 'S', '93 ': 'G', '94 ': 'L', '95 ': 'Q', '96 ': 'S',
'97 ': 'E', '98 ': 'D', '99 ': 'E', '100 ': 'A', '101 ': 'D', '102 ': 'Y', '103 ': 'Y', '104 ': 'R'},
'cdrl3': {'105 ': 'A', '106 ': 'A', '107 ': 'W', '108 ': 'D', '109 ': 'D', '110 ': 'S', '113 ': 'L', '114
': 'N', '115 ': 'G', '116 ': 'W', '117 ': 'V'}, 'fwl4': {'118 ': 'V', '119 ': 'G', '120 ': 'G', '121 ':
'G', '122 ': 'T', '123 ': 'K', '124 ': 'L', '125 ': 'T', '126 ': 'V', '127 ': 'L'}}",|||Missing Conserved
Cysteine: 104||
```

So that the code below is not used in the final version.

```
df = spark.read.text("data.csv")
col_name = (df.tail(2)[0]["value"]).split(",")
data = [df.tail(1)[0]["value"].split(",")]
df1 = spark.createDataFrame(data,col_name)
```

Then filter the data to get the result.

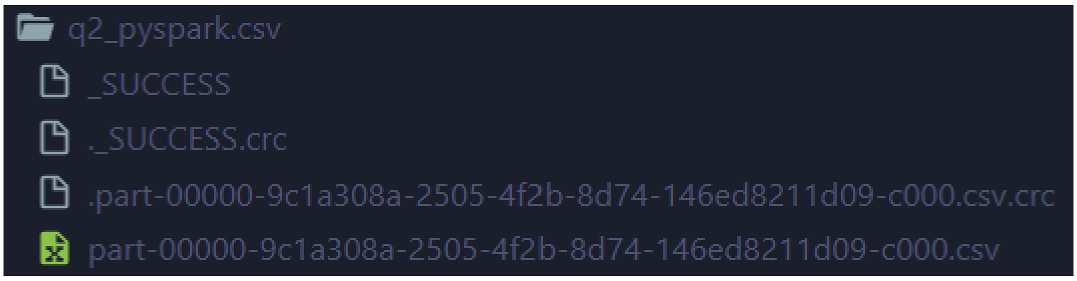
```
df.select(col("sequence_alignment_aa"),col("cdr1_aa"),col("cdr2_aa"),col("cdr3_a
a")).where((length(col("cdr3_aa"))>=10) &  (length(col("cdr3_aa"))<=100 ))
df_remove.coalesce(1).write.format("csv").mode("overwrite").save("output/output_
data/q2_pyspark.csv")
```



```
+--------------------+---------+-------+------------+
|sequence_alignment_aa|  cdr1_aa|cdr2_aa|     cdr3_aa|
+--------------------+---------+-------+------------+
| QSVLTQPPSVSGTPGQR...| DSNIGNNF|    KTS| AAWDDPLNAVL|
| GVPDRFSGSTSGTSASL...|     NULL|   NULL|QSFDNSLGGFYV|
| QSALTQPPSASGTPGQR...| RSNIGINT|    SND| DAWDDNLNGPV|
| QSALTQPASMSGSPGQS...|SSDVGASNH|    EVS|  YSYAVGVTFV|
| TLSLTCGSSTGAVHSGY...|TGAVHSGYY|    STD|LLYFGGIQPLWV|
+--------------------+---------+-------+------------+
only showing top 5 rows
```

Save it to a single `.csv` file (Actually, it generatd 4 files in directory `q2_pyspark.csv`).

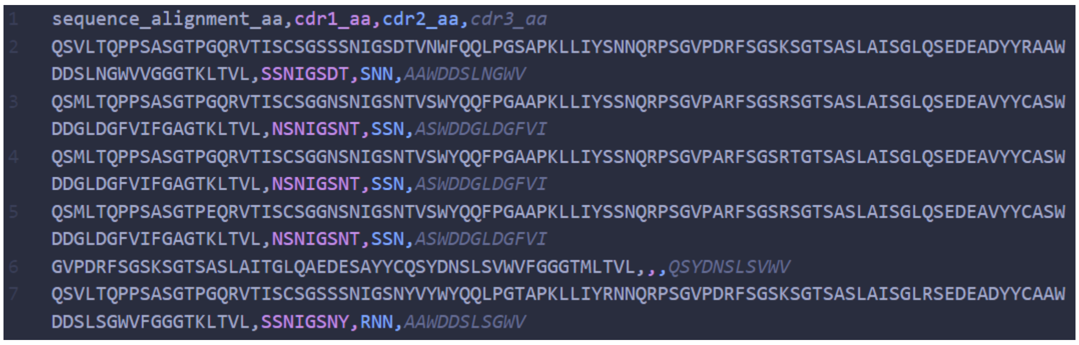
```
df_remove.coalesce(1).write.format("csv").mode("overwrite").save("output/output_data/q2_pyspark.csv")
```



```
q2_pyspark.csv
    _SUCCESS
    ._SUCCESS.crc
    .part-00000-9c1a308a-2505-4f2b-8d74-146ed8211d09-c000.csv.crc
    part-00000-9c1a308a-2505-4f2b-8d74-146ed8211d09-c000.csv
```

(2)

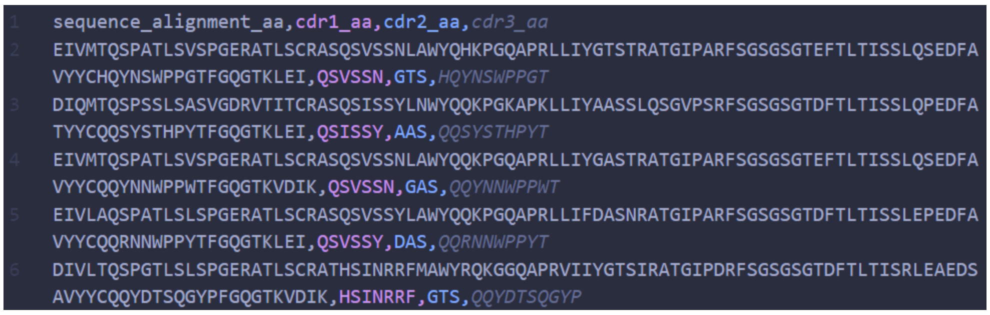
Run the script by command `bash q2_2.sh` and the output file `q2(2)_sh.csv` is generated. There are 17157 rows (including column names) in the output file `q2(2)_sh.csv` .



```
sequence_alignment_aa,cdr1_aa,cdr2_aa,cdr3_aa
QSVLTQPPSASGTPGQRVTISCSGSSSNIGSDTVNWFQQLPGSAPKLLIYSNNQRPSGVPDRFSGSKSGTSASLAISGLQSEDEADYYRAAWDDSLNGWVVGGGTKLTVL,SSNIGSDT,SNN,AAWDDSLNGWV
QSMLTQPPSASGTPGQRVTISCSGGNSNIGSNTVSWYQQFPGAAPKLLIYSSNQRPSGVPARFSGSRSGTSASLAISGLQSEDEAVYYCASWDDGLDGFVIFGAGTKLTVL,NSNIGSNT,SSN,ASWDDGLDGFVI
QSMLTQPPSASGTPGQRVTISCSGGNSNIGSNTVSWYQQFPGAAPKLLIYSSNQRPSGVPARFSGSRTGTSASLAISGLQSEDEAVYYCASWDDGLDGFVIFGAGTKLTVL,NSNIGSNT,SSN,ASWDDGLDGFVI
QSMLTQPPSASGTPEQRVTISCSGGNSNIGSNTVSWYQQFPGAAPKLLIYSSNQRPSGVPARFSGSRSGTSASLAISGLQSEDEAVYYCASWDDGLDGFVIFGAGTKLTVL,NSNIGSNT,SSN,ASWDDGLDGFVI
GVPDRFSGSKSGTSASLAITGLQAEDESAYYCQSYDNSLSVWVFGGGTMLTVL,,,QSYDNSLSVWV
QSVLTQPPSASGTPGQRVTISCSGSSSNIGSNYVYWYQQLPGTAPKLLIYRNNQRPSGVPDRFSGSKSGTSASLAISGLRSEDEADYYCAAWDDSLSGWVFGGGTKLTVL,SSNIGSNY,RNN,AAWDDSLSGWV
```

(3)

Run the script by command `bash q2_3.sh` and the output file `q2(3)_sh.csv` is generated. There are 17140 rows (including column names) in the output file `q2(3)_sh.csv` .



```
sequence_alignment_aa,cdr1_aa,cdr2_aa,cdr3_aa
EIVMTQSPATLSVSPGERATLSCRASQSVSSNLAWYQHKPGQAPRLLIYGTSTRATGIPARFSGSGSGTEFTLTISSLQSEDFAVYYCHQYNSWPPGTFGQGTKLEI,QSVSSN,GTS,HQYNSWPPGT
DIQMTQSPSSLSASVGDRVTITCRASQSISSYLNWYQQKPGKAPKLLIYAASSLQSGVPSRFSGSGSGTDFTLTISSLQPEDFATYYCQQSYSTHPYTFGQGTKLEI,QSISSY,AAS,QQSYSTHPYT
EIVMTQSPATLSVSPGERATLSCRASQSVSSNLAWYQQKPGQAPRLLIYGASTRATGIPARFSGSGSGTEFTLTISSLQSEDFAVYYCQQYNNWPPWTFGQGTKVDIK,QSVSSN,GAS,QQYNNWPPWT
EIVLAQSPATLSLSPGERATLSCRASQSVSSYLAWYQQKPGQAPRLLIFDASNRATGIPARFSGSGSGTDFTLTISSLEPEDFAVYYCQQRNNWPPYTFGQGTKLEI,QSVSSY,DAS,QQRNNWPPYT
DIVLTQSPGTLSLSPGERATLSCRATHSINRRFMAWYRQKGGQAPRVIIYGTSIRATGIPDRFSGSGSGTDFTLTISRLEAEDSAVYYCQQYDTSQGYPFGQGTKVDIK,HSINRRF,GTS,QQYDTSQGYP
```