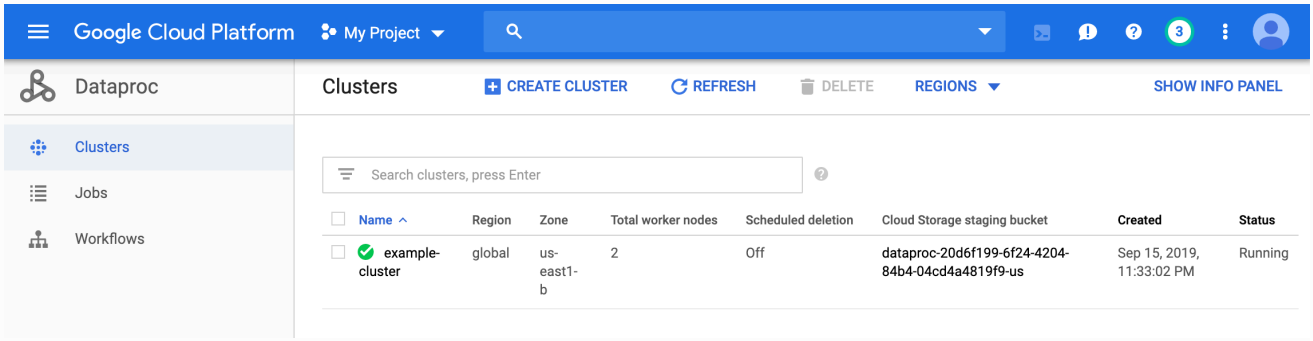
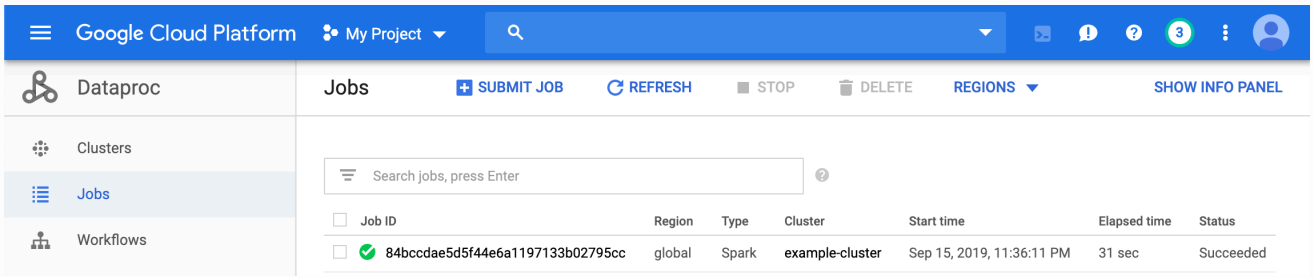
HOMEWORK 0 (E6893)

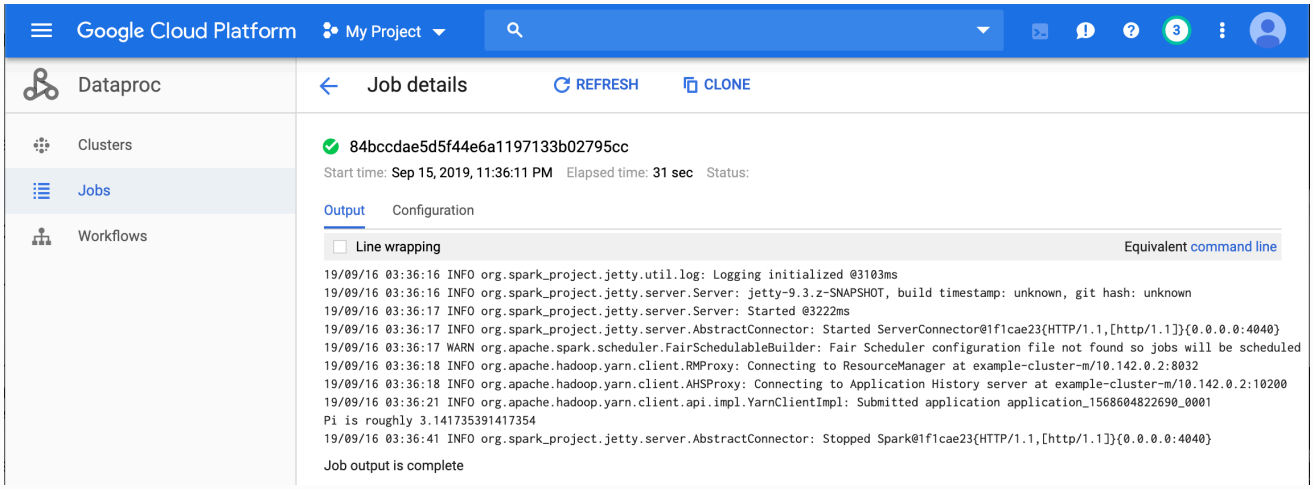
Jing Qian (jq2282)

1. Warm-up exercises

1) Screenshots of exercise 3:







Screenshots of exercise 4:

```
  trackingUrl: http://example-cluster-m:8088/proxy/application_1568604822690_0003/
jq2282@cloudshell:~ (winged-plate-252922)$ gsutil cat gs://example-bucket-jq/output/*
(u'a', 2)
(u'we', 1)
(u'would', 1)
(u"What's", 1)
(u'sweet.', 1)
(u'as', 1)
(u'call', 1)
(u'which', 1)
(u'smell', 1)
(u'name', 1)
(u'That', 1)
(u'rose', 1)
(u'any', 1)
```

2)

Transformations in Exercise3: filter().

Actions in Exercise3: count(). The RDD operation that triggers the program to execute is the actions and hence "count()".

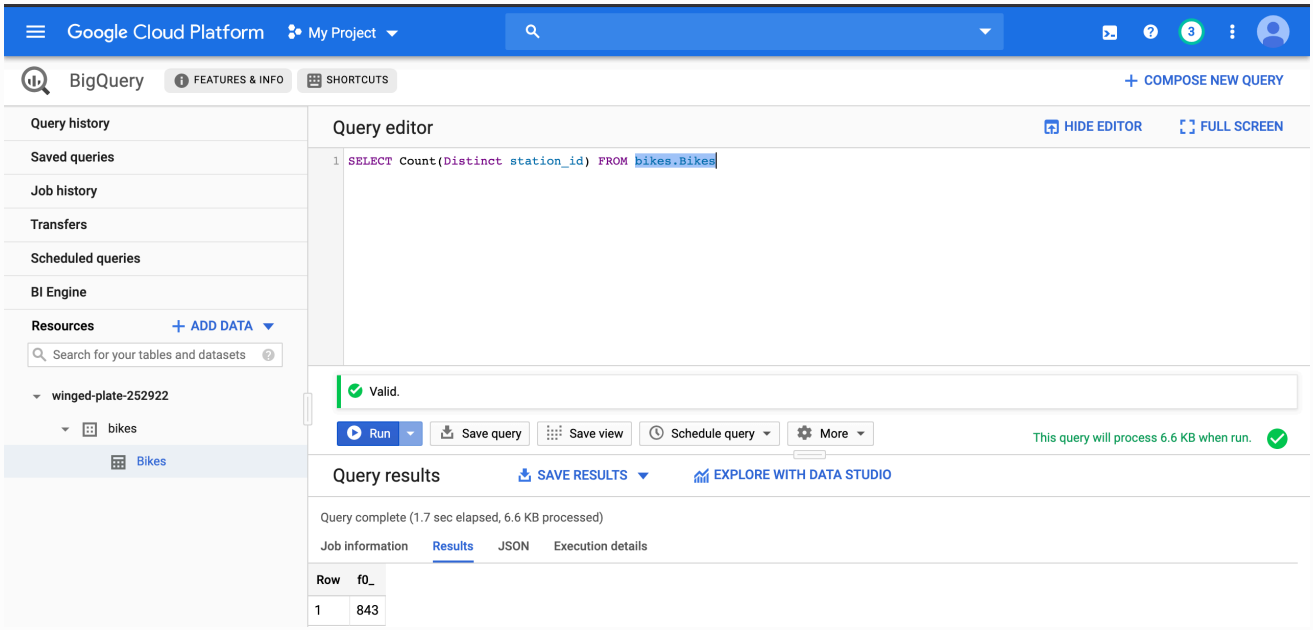
Transformations in Exercise4: flatMap(), map(), reduceByKey().

Actions in Exercise4: saveAsTextFile(). The RDD operation that triggers the program to execute is the actions and hence "saveAsTextFile()".

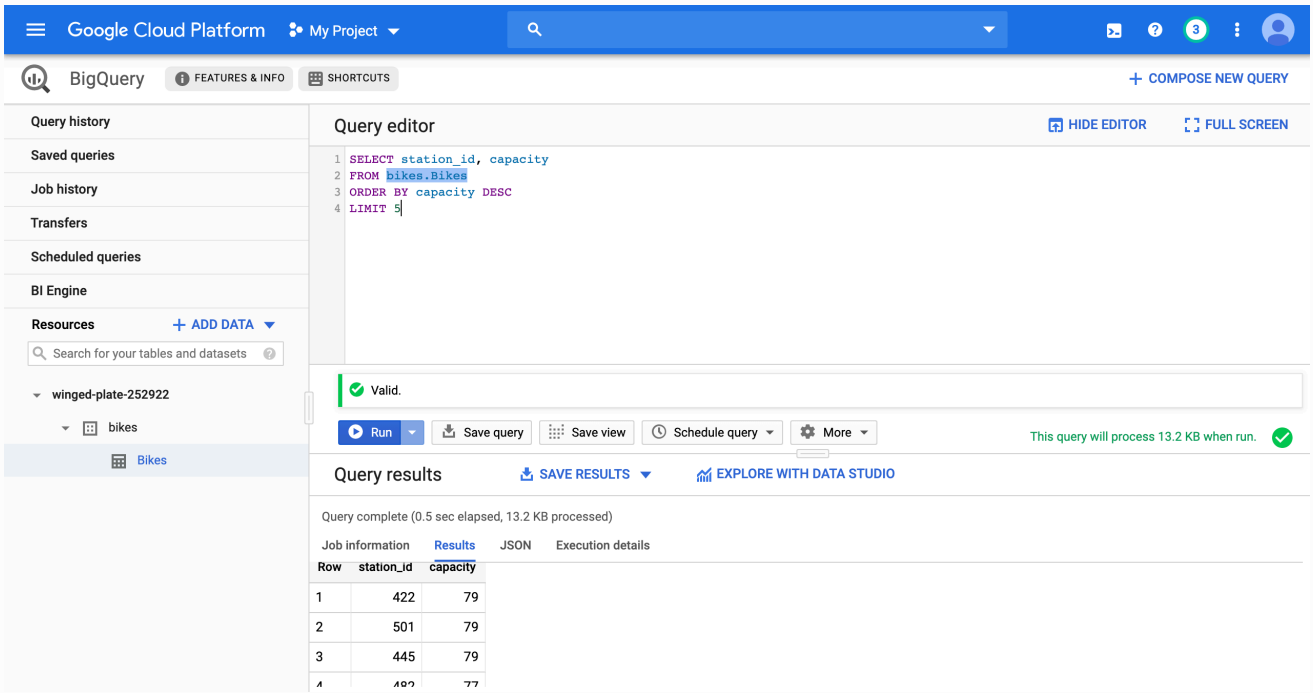
*Exercise 3 is an inside example and corresponding code is found at: https://spark.apache.org/examples.html. The code for Exercise 4 is provided in the given link: https://cloud.google.com/dataproc/docs/tutorials/gcs-connector-spark-tutorial.

2. NYC Bike expert

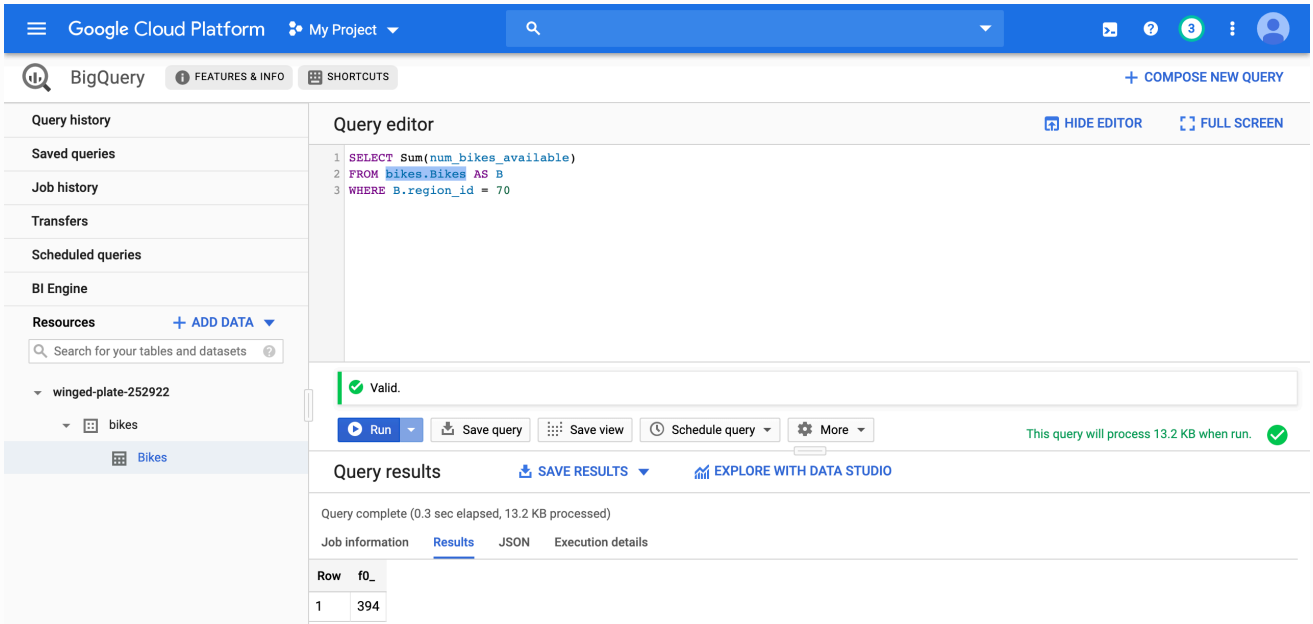
1) There are 843 unique station_ids in this dataset.



2) The largest capacity for a station is 79. The *station_id* of stations that have the largest capacity are: 445, 422, 501.

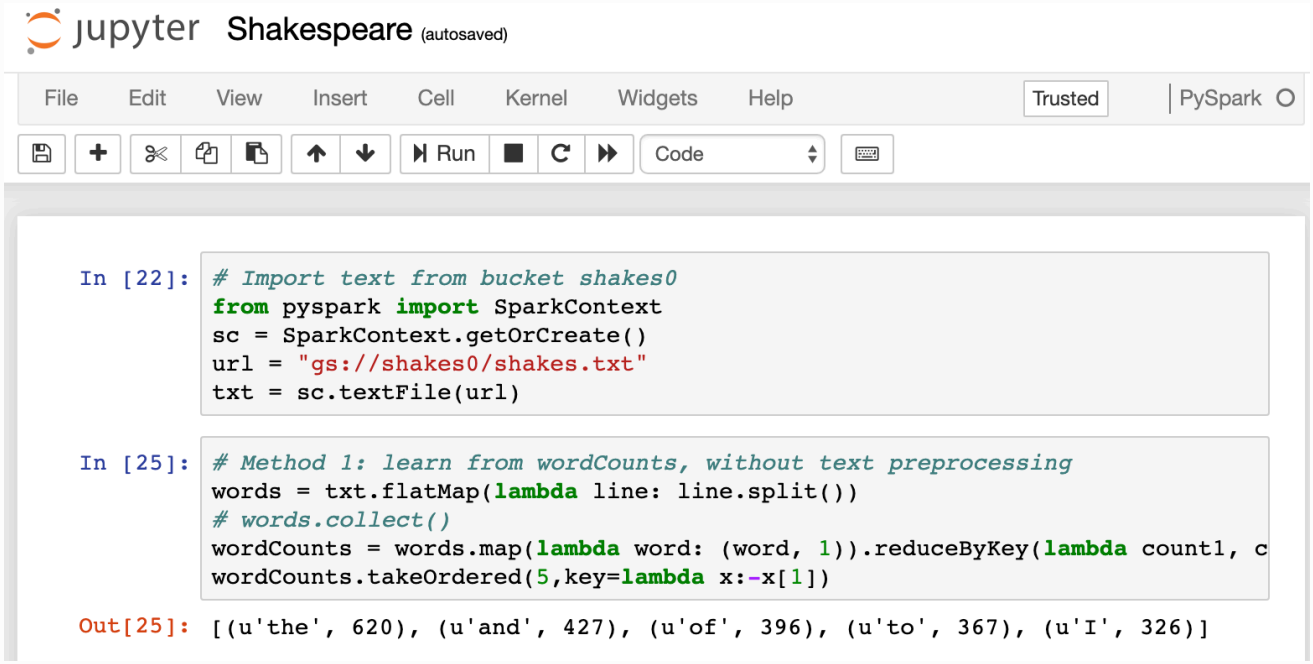


3) The total number of bikes available in region_id 70 is 394.



3. Understanding William Shakespeare

1) The top 5 frequent words without any text preprocessing are: (('the', 620), ('and', 427), ('of', 396), ('to', 367), ('I', 326)).



2) Top 5 frequent words by filtering out stop words provided by NLTK package are: (('I', 346), ('And', 170), ('not', 165), ('with', 143), ('be', 138)). Notice here I also removed the punctuations or the counted words will include punctuations.

jupyter Shakespeare (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted PySpark

Run Code

In [26]:
```
# Method 2: filtering out stop words provided by NLTK package
import nltk

def tokenizer(x):
    from nltk.tokenize import RegexpTokenizer
    token = RegexpTokenizer(r'\w+')
    return token.tokenize(x)

tokens = txt.flatMap(tokenizer)
```

In [27]:
```
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
tokens = tokens.filter(lambda word : word[0] not in stop_words)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

In [28]:
```
wordCount2 = tokens.map(lambda word: (word, 1)).reduceByKey(lambda count1,
wordCount2.takeOrdered(5,key=lambda x:-x[1])
```

Out[28]:
```
[(u'I', 346), (u'And', 170), (u'not', 165), (u'with', 143), (u'be', 138)]
```

The result will differ if we change the splitting method, like following:

In [2]:
```
# Import text from bucket
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
url = "gs://shakespe/shakes.txt"
txt = sc.textFile(url)

# Method 2.2: filter stopwords with nltk but with different splitting metho
words = txt.flatMap(lambda line: line.split())
```

In [3]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
words = words.filter(lambda word : word[0] not in stop_words)
wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda count1, c
wordCounts.takeOrdered(5,key=lambda x:-x[1])
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[3]:
```
[(u'I', 326), (u'And', 169), (u'not', 142), (u'with', 141), (u'Macb.', 13
7)]
```