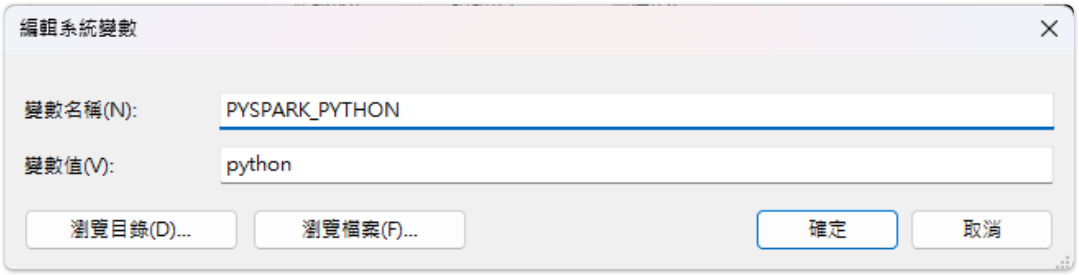
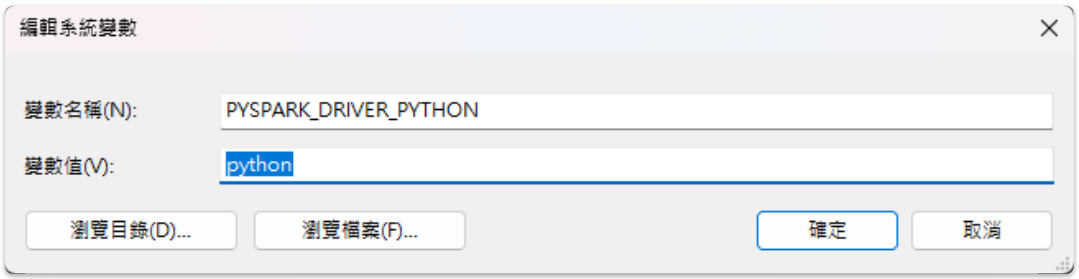
資工碩一 113598007 楊明哲

環境設置

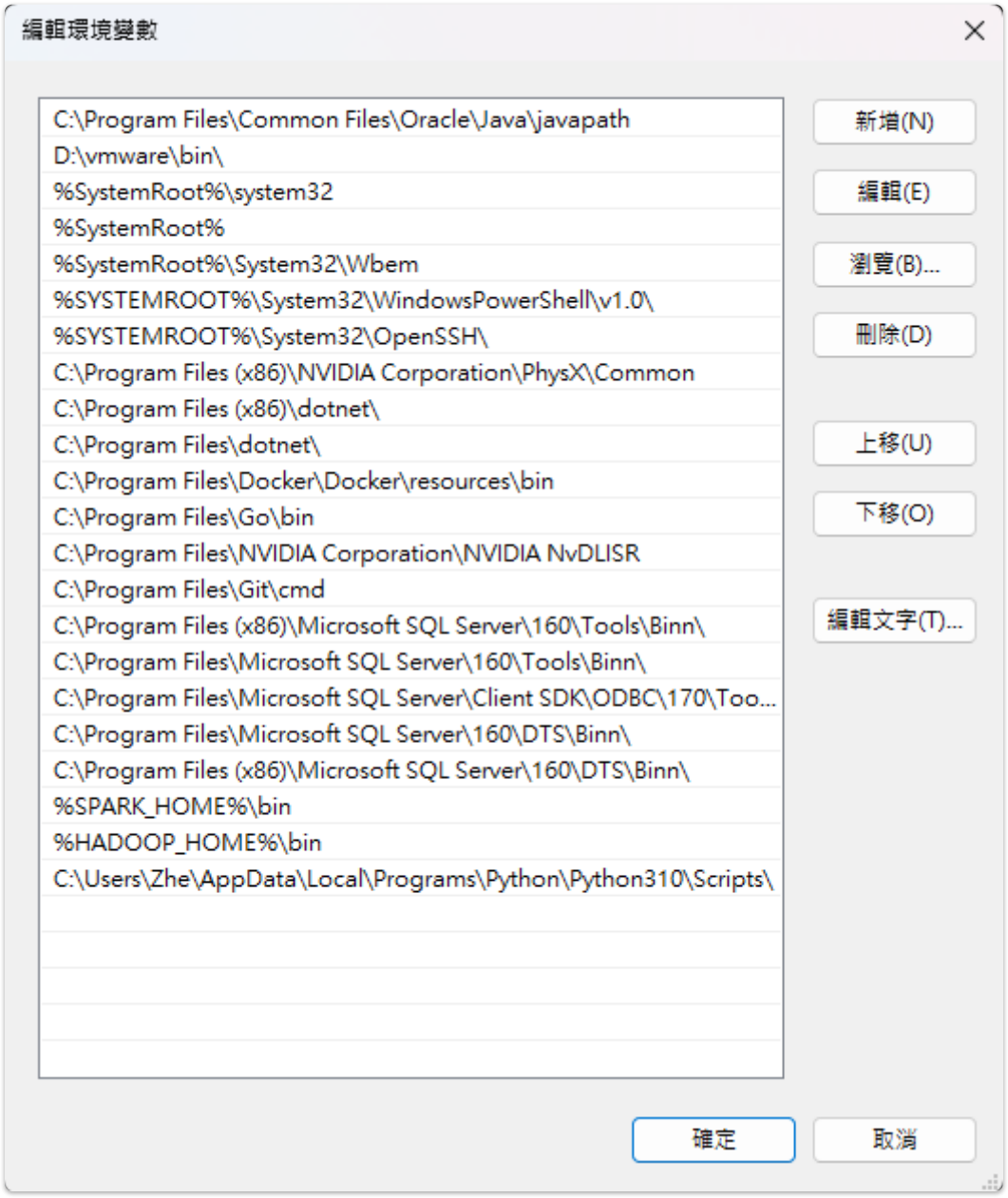
1. Windows
    - 環境版本
        - Java : 17.0.12
        - Python : 3.10.10
        - PySpark : 3.5.3
        - Pandas : 2.2.3
    - 環境配置
        - 下載指定版本的 Python 和 Java
        - 透過 pip install pyspark 安裝 pyspark，並且將它初始化
        - 將下載的 winutils.exe 放進 pyspark 中 bin 的資料夾下
        - 設置環境變數
            - PYSPARK_PYTHON



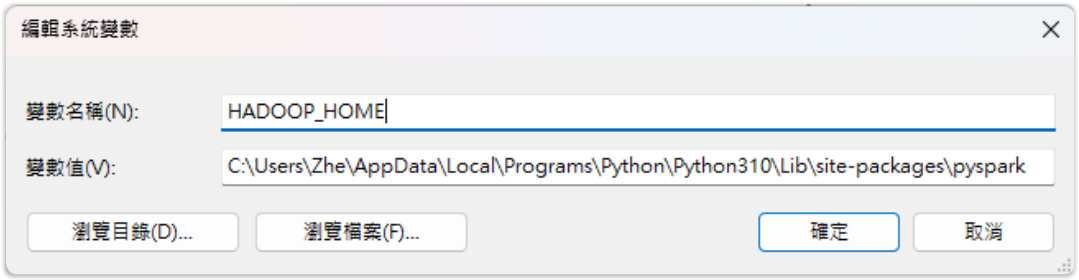
            - PYSPARK_DRIVER_PYTHON



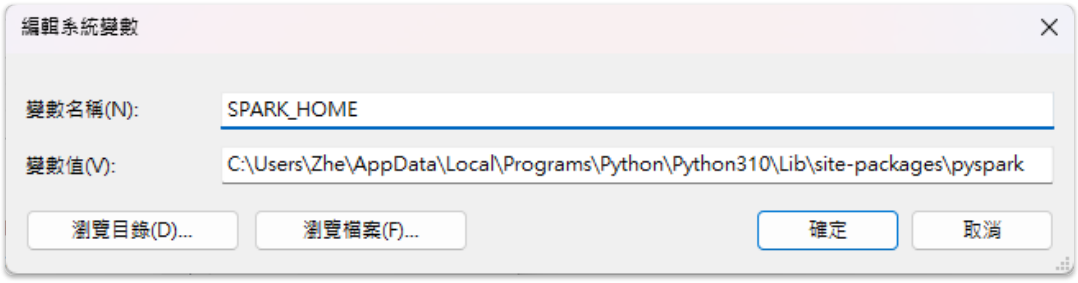
- Path



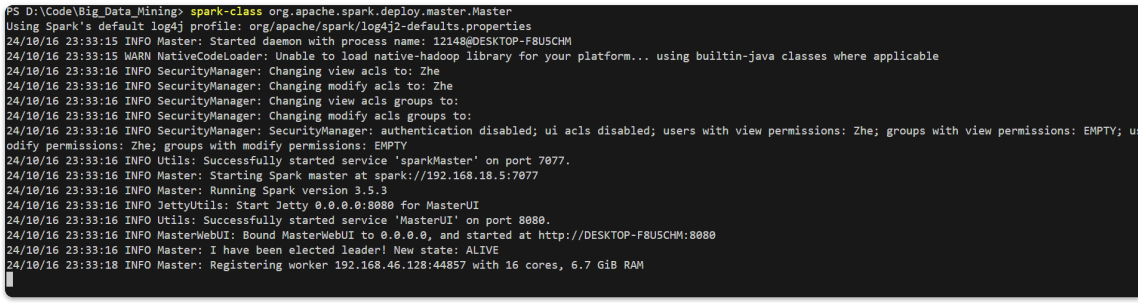
- HADOOP_HOME



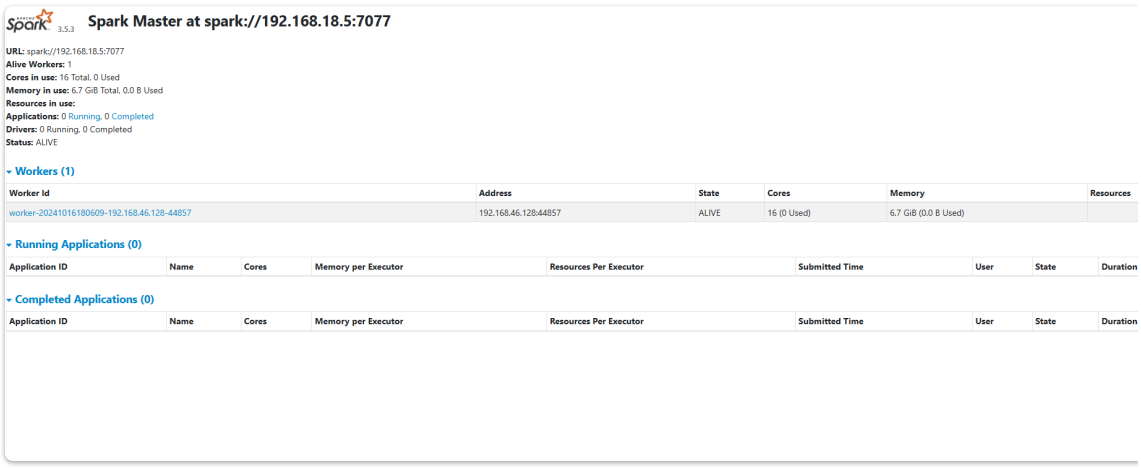
- SPARK_HOME



- 執行指令 spark-class org.apache.spark.deploy.master.Master，並打開 URL
  - 指令結果



  - 網頁結果



2. Ubuntu

- 環境版本
  - Memory : 8GB
  - Processors : 16
  - Java : 17.0.12
  - Python : 3.10.12
  - Spark : 3.5.3
- 環境配置
  - 下載指定的 Java 版本
  - 下載指定的 Spark，並且解壓縮到 Documents 下

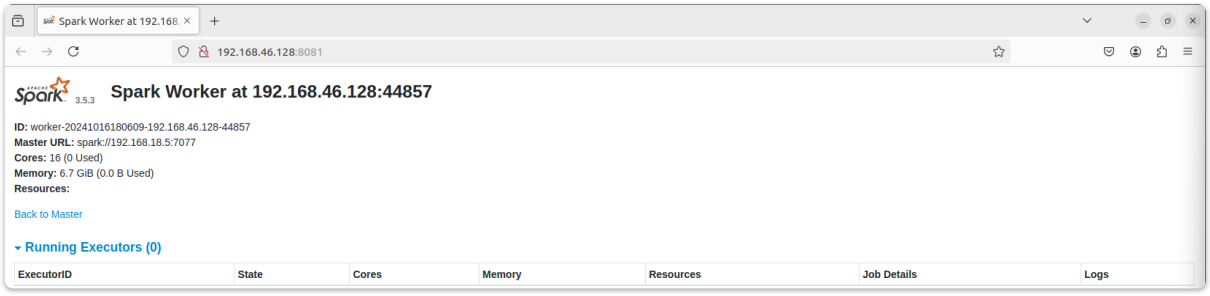
- 將各環境變數設置

```
cluster0@cluster0-virtual-machine: ~
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64

export SPARK_HOME=/home/cluster0/Documents/spark-3.5.3-bin-hadoop3
export HADOOP_HOME=/home/cluster0/Documents/spark-3.5.3-bin-hadoop3

export PATH=$PATH:$SPARK_HOME/bin:$HADOOP_HOME/sbin

# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend
                                                              16,4          Top
```

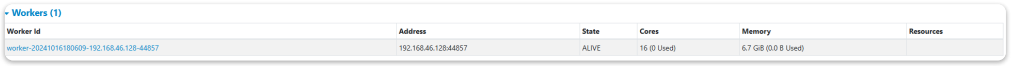
- 執行 spark-class org.apache.spark.deply.worker.Worker spark://192.168.18.5:7077

```
cluster0@cluster0-virtual-machine:~/Documents$ spark-class org.apache.spark.deploy.worker.Worker spark://
192.168.18.5:7077
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
24/10/16 18:06:09 INFO Worker: Started daemon with process name: 19654@cluster0-virtual-machine
24/10/16 18:06:09 INFO SignalUtils: Registering signal handler for TERM
24/10/16 18:06:09 INFO SignalUtils: Registering signal handler for HUP
24/10/16 18:06:09 INFO SignalUtils: Registering signal handler for INT
24/10/16 18:06:09 WARN Utils: Your hostname, cluster0-virtual-machine resolves to a loopback address: 127
.0.1.1; using 192.168.46.128 instead (on interface ens33)
24/10/16 18:06:09 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
24/10/16 18:06:09 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using
builtin-java classes where applicable
24/10/16 18:06:09 INFO SecurityManager: Changing view acls to: cluster0
24/10/16 18:06:09 INFO SecurityManager: Changing modify acls to: cluster0
24/10/16 18:06:09 INFO SecurityManager: Changing view acls groups to:
24/10/16 18:06:09 INFO SecurityManager: Changing modify acls groups to:
24/10/16 18:06:09 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users
 with view permissions: cluster0; groups with view permissions: EMPTY; users with modify permissions: clu
ster0; groups with modify permissions: EMPTY
24/10/16 18:06:09 INFO Utils: Successfully started service 'sparkWorker' on port 44857.
24/10/16 18:06:09 INFO Worker: Worker decommissioning not enabled.
24/10/16 18:06:10 INFO Worker: Starting Spark worker 192.168.46.128:44857 with 16 cores, 6.7 GiB RAM
24/10/16 18:06:10 INFO Worker: Running Spark version 3.5.3
24/10/16 18:06:10 INFO Worker: Spark home: /home/cluster0/Documents/spark-3.5.3-bin-hadoop3
24/10/16 18:06:10 INFO ResourceUtils: ==============================================================
24/10/16 18:06:10 INFO ResourceUtils: No custom resources configured for spark.worker.
24/10/16 18:06:10 INFO ResourceUtils: ==============================================================
24/10/16 18:06:10 INFO JettyUtils: Start Jetty 0.0.0.0:8081 for WorkerUI
24/10/16 18:06:10 INFO Utils: Successfully started service 'WorkerUI' on port 8081.
24/10/16 18:06:10 INFO WorkerWebUI: Bound WorkerWebUI to 0.0.0.0, and started at http://192.168.46.128:80
81
```

- 網頁畫面



- 並且能夠在 Master 上的網頁看到 Worker 的資訊



源代碼

```python
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, min, max, mean, stddev

# Create SparkSession
spark = SparkSession.builder.appName("Electric Power Consumption").getOrCreate()

# Read Data
file_path = "D:/Code/Big_Data_Mining/Electric_Power_Consumption/household_power_consumption.txt"
df = pd.read_csv(file_path, sep=';', low_memory=False)

# Drop missing values
df.dropna(inplace=True)

# Data type conversion
df['Global_active_power'] = df['Global_active_power'].astype(float)
df['Global_reactive_power'] = df['Global_reactive_power'].astype(float)
df['Voltage'] = df['Voltage'].astype(float)
df['Global_intensity'] = df['Global_intensity'].astype(float)

# Convert pandas DataFrame to Spark DataFrame
spark_df = spark.createDataFrame(df)

# Q1: Find min, max, count
min_values = spark_df.select(
    min(col('Global_active_power')).alias('Global_active_power'),
    min(col('Global_reactive_power')).alias('Global_reactive_power'),
    min(col('Voltage')).alias('Voltage'),
    min(col('Global_intensity')).alias('Global_intensity')
).collect()[0]

max_values = spark_df.select(
    max(col('Global_active_power')).alias('Global_active_power'),
    max(col('Global_reactive_power')).alias('Global_reactive_power'),
    max(col('Voltage')).alias('Voltage'),
    max(col('Global_intensity')).alias('Global_intensity')
).collect()[0]
```

```python
count_values = spark_df.count()

stats1 = pd.DataFrame({
    'Min': [min_values['Global_active_power'], min_values['Global_reactive_power'], min_values['Voltage'], min_values['Global_intensity']],
    'Max': [max_values['Global_active_power'], max_values['Global_reactive_power'], max_values['Voltage'], max_values['Global_intensity']],
    'Count': [count_values, count_values, count_values, count_values]
}, index=['Global_active_power', 'Global_reactive_power', 'Voltage', 'Global_intensity'])

print("Q1:")
print(stats1)

# Q2: Find mean, standard deviation
mean_values = spark_df.select(
    mean(col('Global_active_power')).alias('Global_active_power'),
    mean(col('Global_reactive_power')).alias('Global_reactive_power'),
    mean(col('Voltage')).alias('Voltage'),
    mean(col('Global_intensity')).alias('Global_intensity')
).collect()[0]

stddev_values = spark_df.select(
    stddev(col('Global_active_power')).alias('Global_active_power'),
    stddev(col('Global_reactive_power')).alias('Global_reactive_power'),
    stddev(col('Voltage')).alias('Voltage'),
    stddev(col('Global_intensity')).alias('Global_intensity')
).collect()[0]

stats2 = pd.DataFrame({
    'Mean': [mean_values['Global_active_power'], mean_values['Global_reactive_power'], mean_values['Voltage'], mean_values['Global_intensity']],
    'StdDev': [stddev_values['Global_active_power'], stddev_values['Global_reactive_power'], stddev_values['Voltage'], stddev_values['Global_intensity']]
}, index=['Global_active_power', 'Global_reactive_power', 'Voltage', 'Global_intensity'])

print("Q2:")
print(stats2)

# Q3: Min-Max Normalization
normalized_df = spark_df.select(
    ((col('Global_active_power') - min_values['Global_active_power']) / (max_values['Global_active_power'] - min_values['Global_active_power'])).alias('Normalized_Global_active_pow
```

```python
    ((col('Global_reactive_power') - min_values['Global_reactive_power']) / (max_values['Global_reactive_power'] - min_values['Global_reactive_power'])).alias('Normalized_Global_re
    ((col('Voltage') - min_values['Voltage']) / (max_values['Voltage'] - min_values['Voltage'])).alias('Normalized_Voltage'),
    ((col('Global_intensity') - min_values['Global_intensity']) / (max_values['Global_intensity'] - min_values['Global_intensity'])).alias('Normalized_Global_intensity')
)

# Write to CSV
stats1_output_path = "D:/Code/Big_Data_Mining/result/Q1.csv"
stats2_output_path = "D:/Code/Big_Data_Mining/result/Q2.csv"
normalized_output_path = "D:/Code/Big_Data_Mining/result/Q3.csv"

stats1.to_csv(stats1_output_path)
stats2.to_csv(stats2_output_path)
normalized_df.toPandas().to_csv(normalized_output_path, index=False)
```

輸出結果

- 執行 spark-submit --master spark://192.168.18.5:7077 --conf spark.driver.host=192.168.18.5 homework0.py

第一題

- Find min, max, count

```
Q1:
                            Min       Max    Count
Global_active_power       0.076    11.122  2049280
Global_reactive_power     0.000     1.390  2049280
Voltage                 223.200   254.150  2049280
Global_intensity          0.200    48.400  2049280
```

第二題

- Find mean, standard deviation

```
Q2:
                             Mean     StdDev
Global_active_power      1.091615   1.057294
Global_reactive_power    0.123714   0.112722
Voltage                240.839858   3.239987
Global_intensity         4.627759   4.444396
```

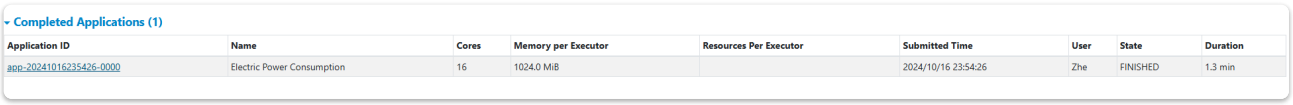
第三題

- Normalization

```
Normalized_Global_active_power,Normalized_Global_reactive_power,Normalized_Voltage,Normalized_Global_intensity
0.3747963063552418,0.30071942446043165,0.376090468497577,0.37759336099585067
0.4783632084012313,0.31366906474820144,0.33699515347334413,0.47302904564315357
0.4796306355241717,0.35827338129496406,0.32600969305331173,0.47302904564315357
0.48089806264711216,0.3611510791366907,0.3405492730210021,0.47302904564315357
0.3250045265254391,0.3798561151079137,0.4032310177705981,0.32365145228215775
0.311787072243346,0.37553956834532376,0.3819063004846531,0.3070539419087137
0.3282636248415716,0.3741007194244605,0.3841680129240713,0.32365145228215775
0.32808256382400874,0.3741007194244605,0.38836833602584825,0.32365145228215775
0.32518558754300203,0.36690647482014394,0.3486268174474964,0.32365145228215775
0.32464240449031323,0.36690647482014394,0.34442649434571954,0.32365145228215775
0.39579938439254037,0.35827338129496406,0.31211631663974215,0.40248962655601667
0.48307079485786714,0.3381294964028777,0.30953150243236355,0.47717842323651455
0.46605105920695283,0.34388489208633094,0.3163166397415191,0.46058091286307057
0.470034401593337,0.2863309352517986,0.3137318255250405,0.4647302904564316
0.36013036393264536,0.303597122302l583,0.3890145395799681,0.3609958506224067
0.29947492304906753,0.2028776978417266,0.4504038772213244,0.29045643153526973
0.28915444504798116,0.10935251798561152,0.43715670436187376,0.28215767634854777
0.30363932645301467,0.11223021582733814,0.4478190630048467,0.29460580912863077
0.28879232301285535,0.0,0.4500807754442649,0.28215767634854777
0.33061741806988896,0.0,0.4084006462035544,0.3360995850622407
0.5267065000905305,0.0,0.3066235864297255,0.5228215767634855
0.690747782002535,0.0,0.25137318255250396,0.6846473029045643
0.6291870360311426,0.0,0.2911147011308567,0.630705394l908715
0.4615245337678799,0.0,0.35508885298869153,0.4522821576763486
0.3981531776208583,0.0,0.37996768982229445,0.39834024896265563
0.2871627738547891,0.0,0.43489499192245556,0.27800829875518673
0.28607640774941157,0.0,0.4084006462035544,0.27800829875518673
0.2853521636791599,0.0,0.4006462035541195,0.27800829875518673
0.28806807894260367,0.0,0.39709208400646245,0.28215767634854777
0.28082563824008694,0.0,0.390306946688207,0.27385892116182575
```

補充

- 網頁顯示結果



Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241016235426-0000	Electric Power Consumption	16	1024.0 MiB		2024/10/16 23:54:26	Zhe	FINISHED	1.3 min