MongoDB
MongoDB

04 Chapter

Aggregatio 進階查詢

4-1 說明

Aggregation 是 MongoDB 非常重要的一項進階查詢功能，可以對原始資料進行拆解、重組與各種數學統計運算。Aggregation 與 find 不同處在於 find 只能對資料進行一次的運算處理，而 aggregation 可以對資料進行多次運算後才取得結果，例如使用 find 可以找出各縣市的 AQI 指數，但是要計算各縣市平均 AQI 指數，find 就無能為力了，這時得要靠 aggregation 才行。

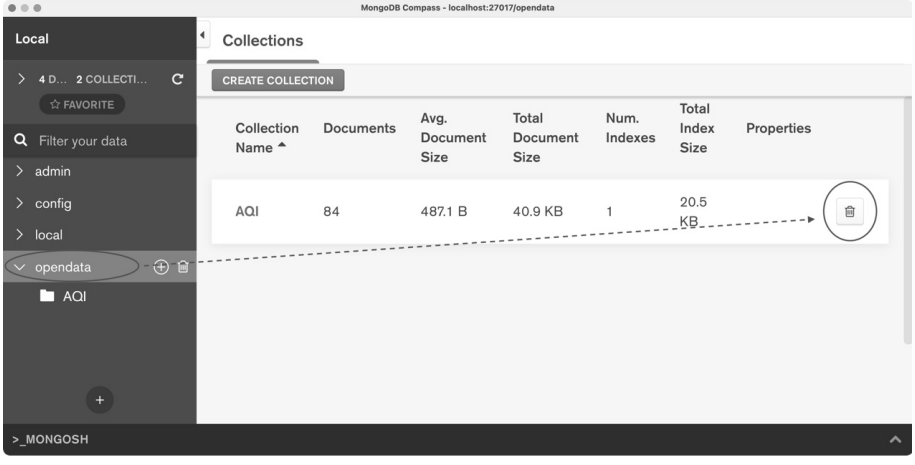
Aggregation 利用 pipeline 方式處理資料。Pipeline 包含了一或數個資料處理階段（稱為 stage），每一個 stage 會將上一個 stage 輸出資料經過處理後傳給下一個 stage，直到最後產生我們要的資料為止。例如在上一章我們從資料開平臺上取得的 AQI 資料，其中各項指數的資料型態都是字串，因此若想要計算 AQI 欄位的平均值，第一個 stage 必須先

將 AQI 型態由字串轉為數字，第二個 stage 才能對同一縣市的各區 AQI 值群組後做平均。所以 pipeline 就是讓資料經過一連串的變化，最後產生我們要的結果。

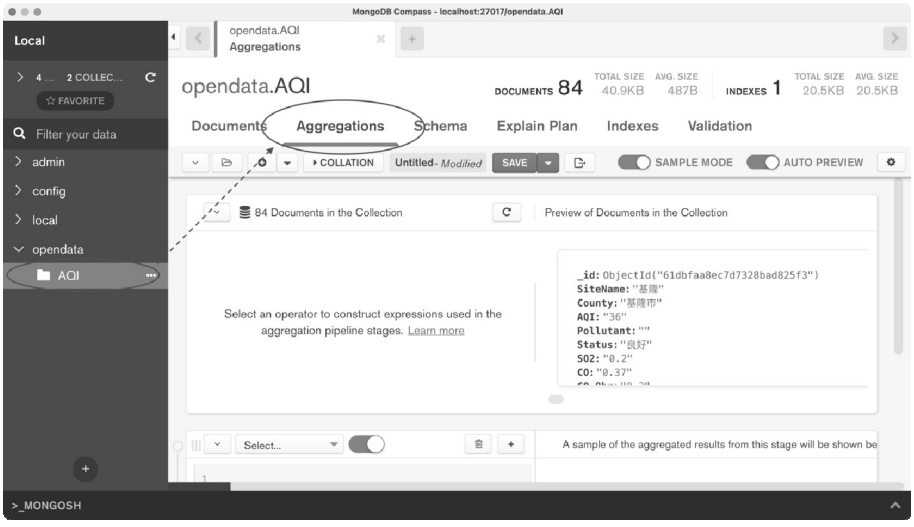
4-2 前置準備

請讀者安裝 MongoDB 官方的 Compass 圖形介面管理系統，我們可以在 Compass 中組合與測試我們的 pipeline，等到一切都正確了，Compass 可以產生對應的 Python 程式碼，複製貼上到 Python 程式中就完成了。透過這個作法來設計 pipeline 會很方便。

請讀者先將 MongoDB 中的 AQI 資料表刪除，因為舊的資料經過前面單元的練習應該已經很凌亂了，重新儲存一份新的 AQI 資料來練習各種 pipeline 操作較為合適。我們可以在 Compass 中刪除資料表，或是在 mongosh 中呼叫 db.opendata.drop() 也可以。



新的 AQI 空氣品質指標資料存入 MongoDB 後，在 Compass 中點選 Aggregations 分頁，可以看到剛新存入的資料，第一列的是原始資料。



在 Aggregation 分頁提供的畫面中，有幾個操作必須先學會。按下「ADD STAGE」或「+」按鈕可以在 pipeline 中新增一個 stage。垃圾桶圖示為刪除 stage。Switch 開關可以 Enable 或 Disable 該 stage。每個 stage 最左上角的「|||」符號拖著上下移動可以改變 stage 順序。

知道這些基本操作後，接下來準備好 Python 範本，如下。其中變數 pipeline 之後會複製貼上從 Compass 過來的程式碼，這裡先填入一個空陣列即可。

```
# Python 程式
import pymongo
from pprint import *

client = pymongo.MongoClient()
db = client.opendata

pipeline = []

cursor = db.AQI.aggregate(pipeline)
pprint(list(cursor))
```

若要在 MongoDB shell 中執行 aggregation，指令與參數與 Python 大同小異，先將指令準備好，函數名稱為 aggregate()，中間的空陣列內容之後會在 Compass 中產生。

```
# MongoDB shell
opendata> db.AQI.aggregate([])
```

4-3 新手必看

這個單元以各縣市平均 AQI 指標為範例，先讓第一次接觸的讀者對 pipeline 運作原理有基本瞭解。本節將會透過四個 stage 產生各縣市的平均 AQI 值，每個小節詳細介紹一個 stage 運作。由於 pipeline 中的各 stage 有順序性，因此接下來四個小節請依序閱讀，每一小節所需要的資料來源為上一小節的輸出。

在進入 pipeline 之前，建議先檢查一下 AQI 這個欄位的值是否有空字串出現，如果有，請將空字串改為 "-1"。會出現空字串的原因是因為該監測站故障，所以產生空字串資料，空字串在轉換成整數型態時會出現錯誤。可以透過下列指令將空字串改為 "-1"。

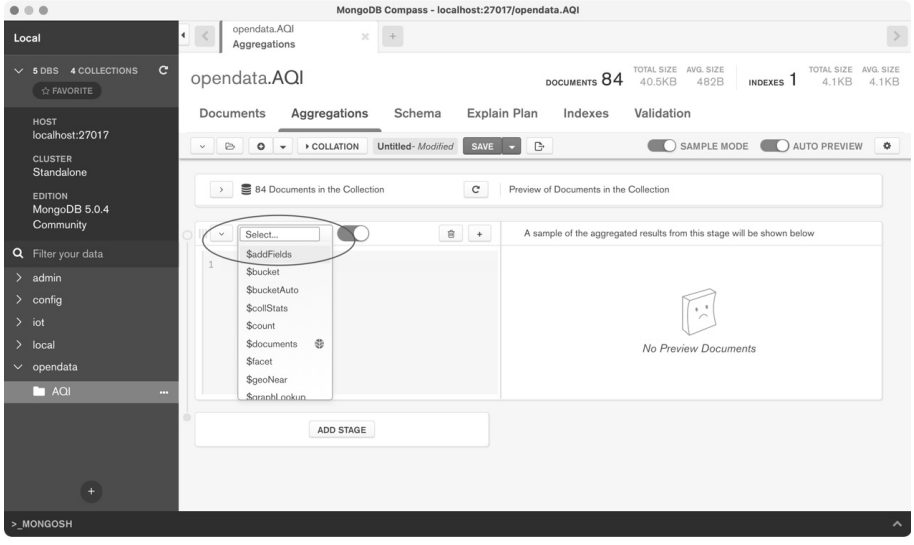
```
# MongoDB shell
opendata> db.AQI.updateMany(
    { 'AQI': '' },
    { '$set': { 'AQI': '-1' }}
)
```

4-3-1 Stage 1 - 新增欄位

新增欄位使用的 stage 名稱為「$addFields」。我們第一個 stage 要將 AQI 欄位的字串型態轉成 Int 型態後放到 iAQI 欄位中，iAQI 就是一個

新增的欄位，是在原本資料表中沒有的，新增完也不會改變原始資料內容，新欄位的資料只暫存於記憶體中，執行完就沒有了。

開啟 Compass 的 Aggregations 分頁，在第一個 stage 的位置（如果畫面上沒有出現的話，按一下下方的「ADD STAGE」按鈕新增即可），然後選擇「$addFields」。



在左側填入下表「$addFields」欄位中內容，若一切正確，在右半邊視窗會立即看到 stage 產出的資料，仔細在右邊找一下，可以發現每筆資料都出現了 iAQI 這個欄位，並且數值為整數型態。我們將右半邊的輸出結果其中一筆放在下表的「輸出結果」欄位中，以截圖方式呈現。

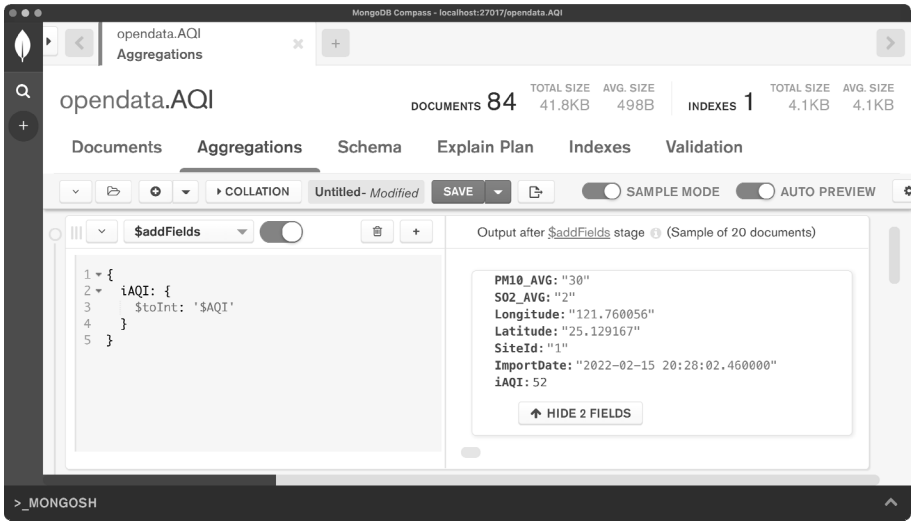
$addFields 內容

`{`
`  iAQI: {`
`  $toInt: '$AQI'`
`  }`
`}`

輸出結果（其中一筆）

```
PM10_AVG: "30"
SO2_AVG: "2"
Longitude: "121.760056"
Latitude: "25.129167"
SiteId: "1"
ImportDate: "2022-02-15 20:28:02.460000"
iAQI: 52
HIDE 2 FIELDS
```

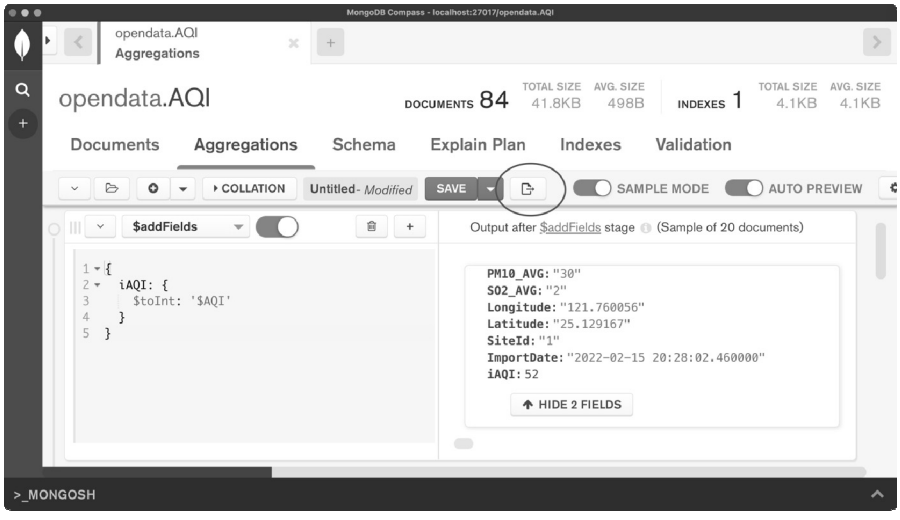
上表顯示的指令與輸出結果在 Compass 的 Aggregations 頁面呈現如下，由於書本印刷不如螢幕截圖來得清晰，因此，之後在 Compass 中每一個 stage 要填的內容都會以上表的方式來呈現。

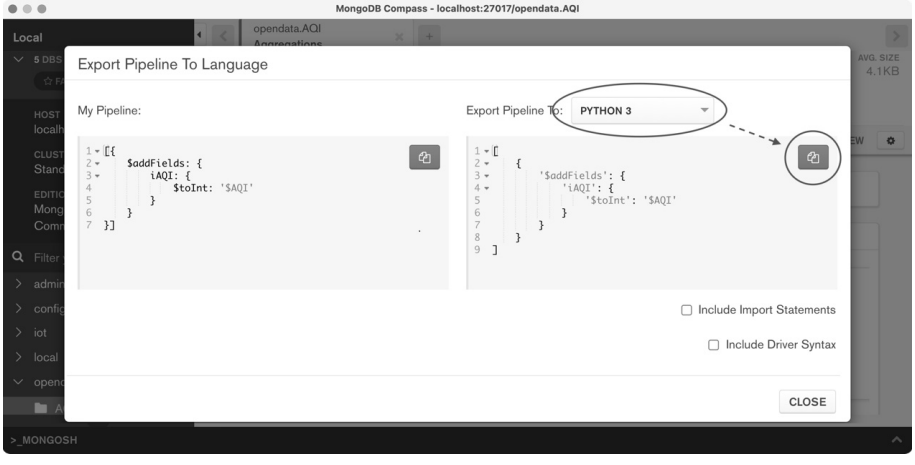


接下來說明「$addFields」填入的內容。iAQI 為新的欄位名稱，可以根據需要隨意修改，名稱前後的單引號或雙引號在這裡省略掉了。「$toInt」為型別轉換運算子，由運算子名稱就知道要轉成整數，前後的單引號或雙引號同樣也省略了。最後的 '$AQI' 表示要轉換成整數的欄位為 AQI，這裡的單引號或雙引號不可以省略，一旦省略，會讓

aggregation 認為 $AQI 是一個內建的系統函數，這當然是錯誤的。此外，'$AQI' 中的「$」也不可以省略，一旦省略，代表要轉成整數的資料來源為字串 AQI，而不是 AQI 欄位內容，除了不會有任何資料產出外，還會得到錯誤訊息，因為字串 AQI 無法轉成整數。

如果右半邊視窗中的資料已經是我們要的結果，接下來點選「匯出」按鈕，然後選擇 Python 3 語言。





將程式碼複製貼上到 Python 中的 pipeline 變數就完成了，完整的程式碼如下。

```
# Python 程式
import pymongo
from pprint import *

client = pymongo.MongoClient()
db = client.opendata

pipeline = [
    {
        '$addFields': {
            'iAQI': {
                '$toInt': '$AQI'
            }
        }
    }
]

cursor = db.AQI.aggregate(pipeline)
pprint(list(cursor))
```

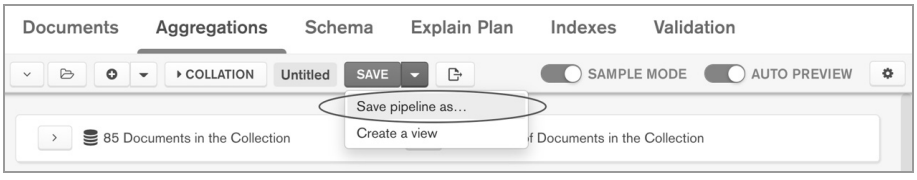
執行看看，輸出結果會出現 iAQI 這個欄位，並且資料型態為整數，如下。

```
# Output
[
    {
        ...
        'SiteName': '基隆',
        'iAQI': 52
    },
    ...
]
```

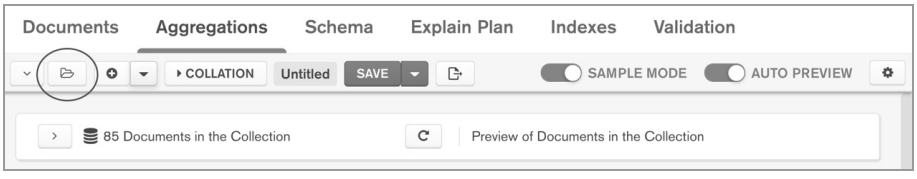
如果我們要在 MongoDB shell 中執行 aggregation 的 pipeline stage 指令，跟 Python 一樣使用 aggregate() 函數，在 Compass 中複製語法時改為複製左邊的 My Pipeline，如下圖。

4-3-6 存成其他格式

如果只是要將 pipeline 儲存起來，但不要變成 View，在前圖的「SAVE」按鈕按下後選擇「Save pipeline as…」，然後給一個名字就可以。

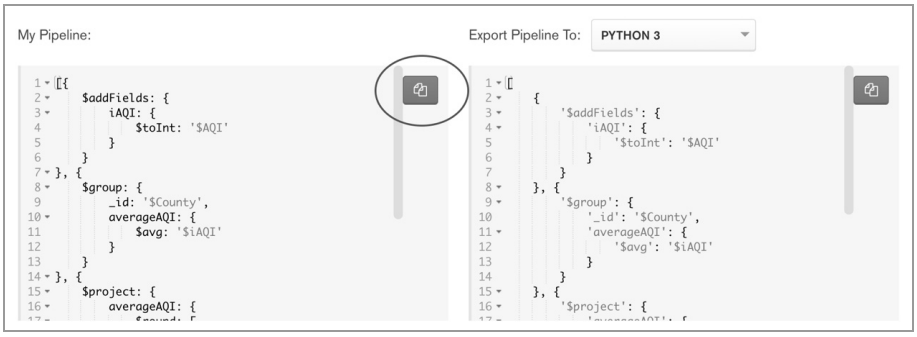


之後點選下圖圈出的按鈕就可以把儲存在 MongoDB 的 pipeline 再叫出來編輯或執行。

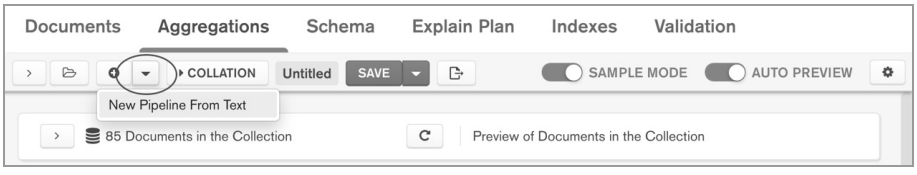


也可以點選下圖按鈕選擇存成外部的文字檔。

點選下圖左側的按鈕，將複製下來的程式碼貼到某個文字編輯器並存檔。



如果要讀取外部存檔的內容，只要用文字編輯器開啟外部檔案，然後複製貼上到「＋」號右邊的下拉箭頭，然後選擇「New Pipeline From Text」就可以了。



4-4 常用 Stage 介紹

4-4-1 桶型計算

「$bucket」用於桶型計算。桶型計算就是將資料按照特定範圍進行群組，通常用在數字型態的資料，例如計算全班成績 0-10 分有幾人、11-20 分有幾人、21-30 分有幾人…等，這就是常見的一種桶型計算。

這裡使用 AQI 資料做為來源資料，並且按照 AQI 數值所定義的污染程度來群組資料，需要兩個 stage 來完成。

Stage 1 選擇 opendata.AQI 資料表，將字串型態的 AQI 轉成整數型態放到 iAQI 欄位，此外，順便透過「$cond」條件判斷運算子

將故障觀測站的 AQI 值轉成 -1，原本是空字串。由於偶爾會有觀測站故障的情況，要特別注意此種情形。

$addFields 內容

```
{
  iAQI: {
    $cond: {
      if: { $eq: ['$AQI', ''] },
      then: -1,
      else: { $toInt: '$AQI' }
    }
  }
}
```

輸出結果（其中一筆）

```
PM10_AVG: "30"
SO2_AVG: "2"
Longitude: "121.760056"
Latitude: "25.129167"
SiteId: "1"
ImportDate: "2022-02-15 20:28:02.460000"
iAQI: 52
```

HIDE 2 FIELDS

Stage 2 根據 AQI 的污染程度分類（網路上查的到），進行桶型計算。

$bucket 內容

```
{
  groupBy: '$iAQI',
  boundaries: [0, 51, 101, 151, 201, 301, 1000],
  default: 'error',
  output: {
    count: { $sum: 1 },
    location: {
```

```
      $push: {
        County: '$County',
        SiteName: '$SiteName',
        iAQI: '$iAQI'
      }
    }
  }
}
```

輸出結果（其中一筆）

```
  _id: 0
  count: 16
▼ location: Array
  ▼ 0: Object
      County: "新北市"
      SiteName: "土城"
      iAQI: 43
  ▼ 1: Object
      County: "新北市"
      SiteName: "板橋"
      iAQI: 50
  ▶ 2: Object
  ▶ 3: Object
```

「$bucket」參數頗多，但不難理解。groupBy 是決定要用哪個欄位進行群組；boundaries 設定群組區段，後面的陣列元素必須遞增排序，並且資料型態要一致。陣列中元素與元素間形成範圍區段，每個區段不包含結尾值，例如 0, 51 表示包含 0 但不包含 51，最後的 1000 隨意設定一個很大的數字即可。MongoDB 在最後一個數字其實可以填保留字 infinite，但因為無法轉成 Python 程式，所以這裡改填一個很大的數字。default 表示不在範圍內的其他數字，如果監測站故障，AQI 為-1 的資料就會落在 default 所指定的欄位中，所以 error 表示欄位名稱。參數 output 是希望輸出的資料，這裡輸出兩個欄位 count 與 location，欄位名稱可以自訂。欄位 count 用來計算該區間中有幾筆資料；欄位 location 透過「$push」運算子將原始資料的三個欄位放到陣列中。參數 output

可以省略，如果省略，會自動產生欄位 count，存放每個區段有多少筆資料。

4-4-2 資料筆數

「$count」用來計算上一個 stage 輸出的資料筆數。只要給一個欄位名稱即可，例如 total。選擇 opendata.AQI 資料表，範例如下。

$count 內容
`'total'`
輸出結果
total: 84

如果想要計算每個群組中的資料筆數，不能$group 後使用$count，因為這兩個同屬於 pipeline stage，所以不可以放在同一個 stage 中使用，必須使用$sum 運算子。例如以 AQI 資料為例，若要計算每個縣市有多少空氣品質監測站，必須根據 County 欄位群組後使用$sum 計算每個群組有幾筆資料，如下。輸出結果代表每個縣市有多少監測站，例如臺北市有 7 個監測站。

$group 內容

```
{
  _id: '$County',
  count: {
    $sum: 1
  }
}
```

輸出結果（其中一筆）

```
_id: "臺北市"
count: 7
```

4-4-3 依經緯度排序

「$geoNear」用在根據經緯度座標產生由近到遠的資料排序，使用這個 stage 有幾個條件：

1. 經緯度資料必須符合 GeoJSON 格式，點座標先放經度再放緯度，例如 [Longitude, Latitude]。
2. 資料表必須建立經緯度座標索引。
3. 必須為 pipeline 的第一個為 stage。
4. 無法在 View 上使用。

檢視目前的 AQI 資料表雖然每筆資料已經包含了經緯度資訊，但不符合 GeoJSON 格式，所以要重新儲存一份符合 GeoJSON 格式的資料。可以用兩種方式將經緯度座標調整到正確格式，一種是從抓政府資料開放平臺的 Python 程式下去改，讓資料存進資料庫時就符合格式；另一種則是透過幾個 pipeline stage 建立一份符合經緯度格式的新資料表。這裡我們選擇後者，可以多一次熟悉 pipeline stage 的操作機會。總共需要三個 stage 來完成這個格式調整的工作。

Stage 1 選擇 opendata.AQI 資料表，將 AQI 中的經緯度欄位內容轉成 GeoJSON 格式。在 GeoJSON 中的欄位 coordinates 陣列先存放經度再存放緯度， 欄位 type 內容為 Point 表示 coordinates 的內容為一個座標點。GeoJSON 還有別的 type，例如 LineString 表示直線，或是 Polygon 表示幾何圖形…等，這部分請自行參考官方文件（https://docs.mongodb.com/manual/reference/geojson/）。

$addFields 內容

```
{
  geometry: {
    type: 'Point',
    coordinates: [
      { $toDouble: '$Longitude' },
      { $toDouble: '$Latitude' }
    ]
  }
}
```

輸出結果（其中一筆）

```
  SiteId: "1"
  ImportDate: "2022-02-15 20:28:02.460000"
▼ geometry: Object
    type: "Point"
  ▼ coordinates: Array
      0: 121.760056
      1: 25.129167
```

↑ HIDE 2 FIELDS

Stage 2 為方便觀察，留下我們需要的欄位即可，其他欄位暫時移除。最後只輸出 SiteName、County、AQI 與 geometry 這四個欄位，順便將 AQI 轉成數字型態。

$project 內容

```
{
  _id: 0, County: 1, SiteName: 1,
  geometry: 1,
  AQI: { $toInt: '$AQI' }
}
```

輸出結果（其中一筆）

```
  SiteName: "基隆"
  County: "基隆市"
▼ geometry: Object
    type: "Point"
  ▼ coordinates: Array
      0: 121.760056
      1: 25.129167
  AQI: 52
```

Stage 3 將資料儲存至另一個資料表，這裡命名為 AQI_geo。

$out 內容

```
'AQI_geo'
```

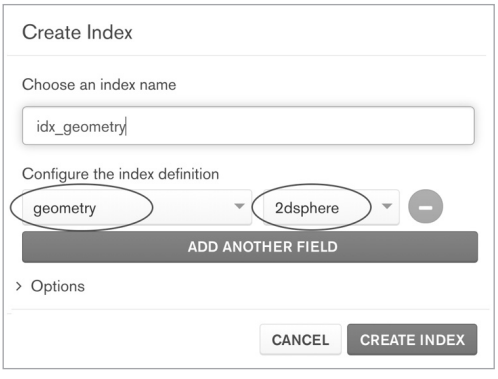
輸出結果

Documents will be saved to the collection: 'AQI_geo'

The $out operator will cause the pipeline to persist the results to the specified location (collection, S3, or Atlas). If the collection exists it will be replaced. Please confirm to execute.

SAVE DOCUMENTS

點選第三個 stage 輸出結果的「SAVE DOCUMENTS」按鈕後重新整理一下 Compass 畫面，應該會看到 AQI_geo 資料表。點選 AQI_geo 資料表，然後在 geometry 欄位上建立型態為 2dsphere 索引（請參考第 8 章）。



索引建立完畢後點選 Aggregations 頁面，現在可以針對 AQI_geo 資料表的經緯度資訊進行排序了，如下。

$geoNear 內容

```
{
  near: {
    type: 'Point',
    coordinates: [121.5466, 25.15532]
  },
  distanceField: 'distance',
  maxDistance: 5000,
  includeLocs: 'geometry',
  query: {
    County: { $in: ['臺北市', '新北市'] }
  }
}
```

輸出結果

```
  _id: ObjectId("620bb060a4825d55ea1b2cea")
  SiteName: "陽明"
  County: "臺北市"
▸ geometry: Object
  AQI: 49
  distance: 3499.1523864049664
```

「$geoNear」的參數很多，說明如下。參數 near 為基準點座標，例如使用者目前所在的位置，最後的查詢結果會以距基準點的距離由近到遠進行排序。以上表的 [121.5466, 25.15532] 為例，此座標為陽明山遊客中心，可以在 Google 地圖中查到此座標點，如右圖。

參數 distanceField 存放與基準點間的距離，也就是遊客中心到這個監測站的距離，欄位名稱 distance 可自訂，單位為公尺，請對照輸出結果的最後一行。參數 maxDistance 設定搜尋最大範圍，單位為公尺，這裡設定搜尋距離遊客中心 5 公里內的資料，出來結果只有一筆，若改為 10 公里，則會出現三筆資料，分別為陽明、士林與淡水。參數 includeLocs 存放原本的經緯度座標資訊，欄位名稱可自訂，此參數為可選參數，意思是不加也可以，請對照輸出結果的 geometry 欄位。參數 query 用來設定搜尋條件，以目前查詢為例，只搜尋臺北市與新北市資料，若省略此參數代表搜尋全部資料。

下面為此 aggregate 在 Python 或 MongoDB shell 中執行後的輸出結果，距離陽明山遊客中心 10 公里範圍內的 AQI 監測站資料，並且由近到遠做排序。

```
# Output
[
  {
    _id: ObjectId("620bb060a4825d55ea1b2cea"),
    SiteName: '陽明',
    County: '臺北市',
    AQI: 49,
    geometry: { type: 'Point', coordinates: [ 121.529583, 25.182722 ] },
    distance: 3499.1523864049664
  },
  {
    _id: ObjectId("620bb060a4825d55ea1b2cb5"),
    SiteName: '士林',
    County: '臺北市',
    AQI: 44,
    geometry: { type: 'Point', coordinates: [ 121.515389, 25.105417 ] },
    distance: 6383.870946549611
  },
  {
    _id: ObjectId("620bb060a4825d55ea1b2cb4"),
    SiteName: '淡水',
```