

Power saver

김영상 백준원 한승헌

2023 06 12

목차

1. 목적
2. 데이터 소개
3. 주요기능
4. 쿼리
5. 인덱싱
6. 시연

목적

영상

내 주위에 사는 사람들에 비해 내가 얼마만큼의 전력량을 사용하는지, 어느 도시가 어떤 목적으로 전력을 가장 많이 소비하는지 알고싶어.



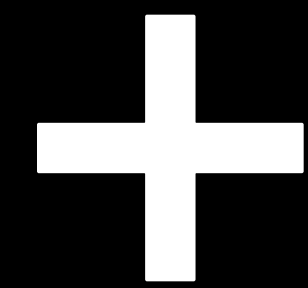
저는 당신 주변 사람들의 전력 사용량이나 개별적인 정보를 알 수 없습니다. 그러나 전력 소비에 대한 일반적인 정보를 제공할 수 있습니다.



도시별로 전력 소비는 여러 요소에 따라 달라집니다. 일반적으로 인구 밀도, 산업 구조, 기후 조건 등이 전력 소비에 영향을 미칠 수 있습니다. 예를 들어, 인구가 많은 도시는 주택, 상업 시설, 교통 등의 전력 수요가 더 커지기 때문에 전력 소비량이 상대적으로 많을 수 있습니다.

또한, 어느 도시가 가장 많은 전력을 소비하는지에 대한 정보는 도시의 산업 구조와 용도에 따라 다를 수 있습니다. 일부 도시는 고도의 산업 부문을 가지고 있어 산업용 전력 소비가 많을 수 있고, 다른 도시는 주거용 전력 소비가 더 많을 수 있습니다. 또한, 목적으로 전력을 가장 많이 소비하는 지역은 국가나 지역의 에너지 정책, 산업 구조, 기후 조건 등에 따라 다를 수 있습니다.

데이터 소개



Location Data

데이터 소개



405.1k
DOCUMENTS

year	조회연도	STRING	조회연도(YYYY)
month	조회월	STRING	조회월(MM)
metro	시도명	STRING	광역자치단체 명(ex:서울특별시, 경기도, 부산광역시 등)
city	시군구명	STRING	기초자치단체 명(ex:청주시, 양평군, 강남구 등)
cntr	계약종별	STRING	한국전력 전기계약종별(ex:주택용, 일반용 등)
custCnt	고객호수	NUMBER	전기사용 고객 호수(호)
powerUsage	전력사용량	NUMBER	전력사용량 합계(kWh)
bill	전기요금	NUMBER	전기요금 합계(원)
unitCost	평균판매단가	NUMBER	전기요금 평균판매 단가(원/kWh)

```
{
  "year": "2020",
  "month": "11",
  "metro": "서울특별시",
  "city": "종로구",
  "cntr": "주택용",
  "custCnt": 65209,
  "powerUsage": 20468684,
  "bill": 2366461741,
  "unitCost": 115.6
}
```

데이터 소개

```
{  
  "City": "강동구",  
  "Latitude": 37.53,  
  "Longitude": 127.1237  
},  
{  
  "City": "강릉시",  
  "Latitude": 37.7525313,  
  "Longitude": 128.8759523  
},  
{  
  "City": "강북구",  
  "Latitude": 37.6395,  
  "Longitude": 127.0255  
},  
{  
  "City": "강서구",  
  "Latitude": 37.5509,  
  "Longitude": 126.8497  
},  
{  
  "City": "강진군",  
  "Latitude": 34.641917,  
  "Longitude": 126.7669591  
},
```

Location Data

210
DOCUMENTS

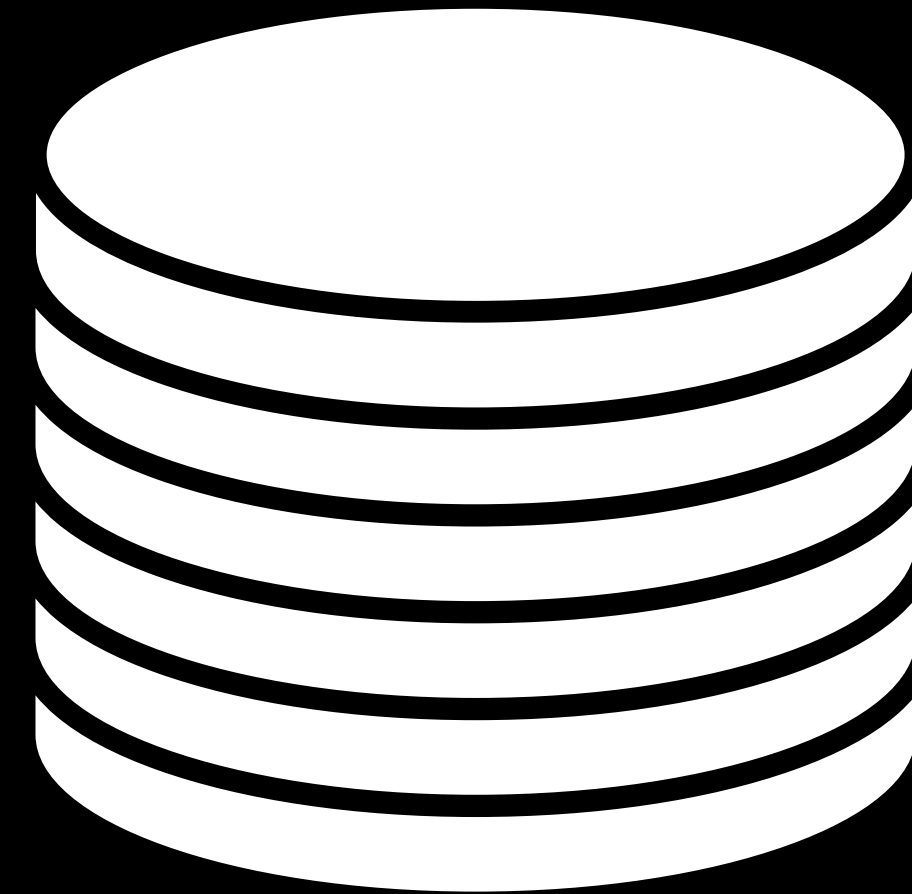
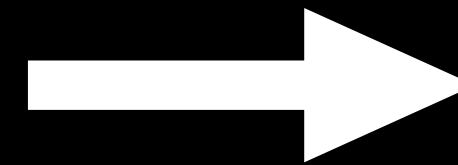
데이터 import

```
with open('/user/city_coordinates/city_coordinates.json', 'r') as file:  
    data = json.load(file)
```

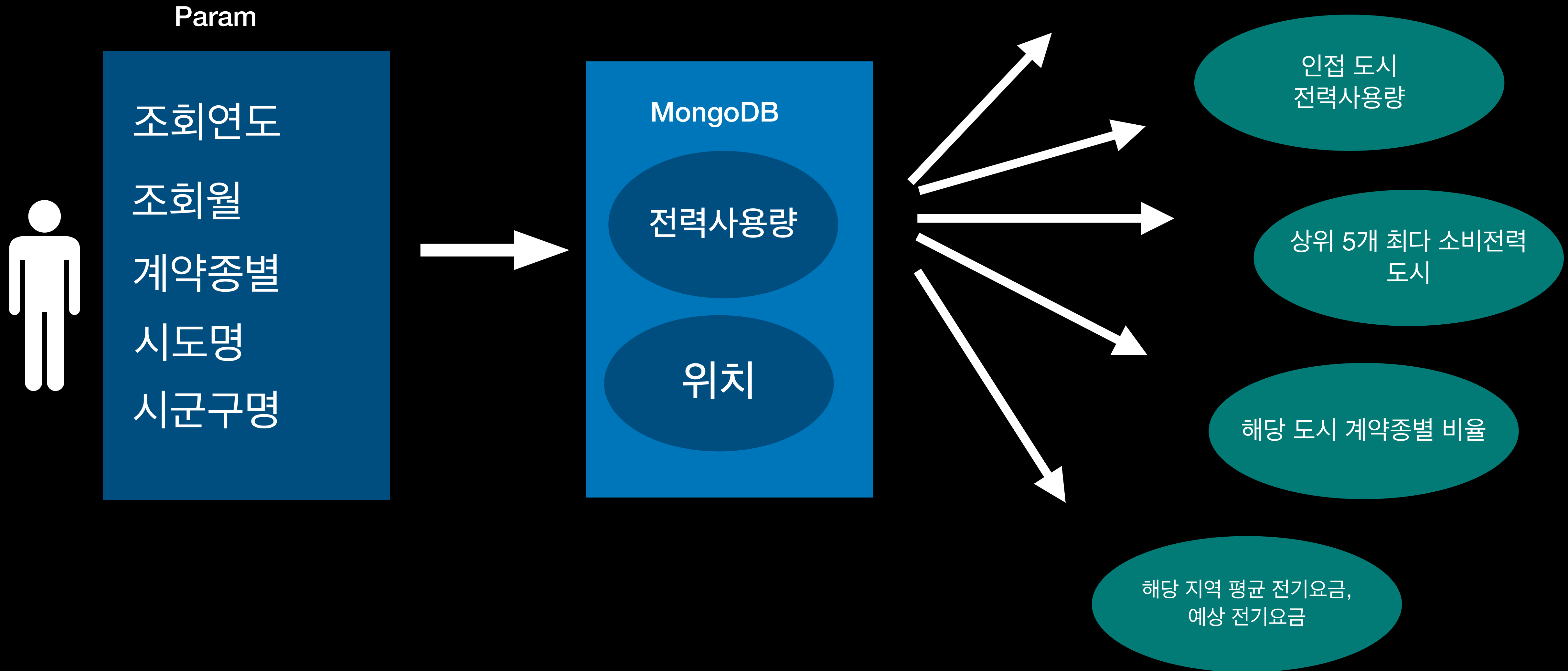
```
# # 데이터를 MongoDB에 저장합니다.  
city_collection.insert_many(data)
```

```
# 2002년부터 2022년까지 데이터 가져오기  
for year in range(2002, 2023):  
    for month in range(1, 13):  
        # API 호출  
        params = {  
            "returnType": "json",  
            "apiKey": api_key,  
            "year": format(year, '02'),  
            "month": format(month, '02')  
        }  
        response = requests.get(base_url, params=params)  
        # 불필요한 공백 및 줄바꿈 문자 제거  
        response_text = response.text.strip()  
  
        # JSON 형식으로 변환  
        data = clean_json_data(response_text)  
        items = data[1]["data"]  
        if data is None:  
            print(f"Error processing data for year {year} and month {month}")  
            continue  
  
        for item in items:  
            collection.insert_one(item)  
        # 진행 상황 출력  
        print(f>Data imported for year {year} and month {month}")
```

load.py



기능



쿼리

해당지역 평균전력사용량

```
pipeline = [  
  {  
    "$match": {  
      "city": city,  
      "metro": metro,  
      "cntr": cntr,  
      "year": year,  
      "month": month  
    }  
  },  
  {  
    "$group": {  
      "_id": None,  
      "average_power_usage": {  
        "$avg": {  
          "$divide": ["$powerUsage", "$custCnt"]  
        }  
      }  
    }  
  }  
]
```

쿼리

인접지역 평균전력사용량

```
nearest_cities = list(self.collection.find(
    {
        "City": {"$ne": target_city}
    }
))
target_city = list(self.collection.find(
    {
        "City": target_city
    }
))[0]

for city in nearest_cities:
    city_lat = city['Latitude']
    city_lon = city['Longitude']
    distance = self.calculate_distance(target_city['La
    distances.append((city["City"], distance))

# 거리를 기준으로 오름차순 정렬
distances.sort(key=lambda x: x[1])

# 최단거리의 도시 추출
nearest_cities = [city[0] for city in distances[:5]]
return nearest_cities
```

```
nearest_city_list = cityCoordinateService.get_nearest_cities_by_geo(city)
nearest_city_list.append(city)
pipeline = [
    {
        '$match': {
            'city': {'$in': nearest_city_list},
            'metro': metro,
            'cntr': cntr,
            'year': year,
            'month': month
        }
    },
    {
        '$group': {
            '_id': "$city",
            'averagePowerUsage': {
                '$avg': {'$divide': ['$powerUsage', '$custCnt']}
            }
        }
    },
    {
        '$project': {
            '_id': 0,
            'name': '$_id',
            'averagePowerUsage': 1
        }
    }
]
```



쿼리

상위 5개 도시

```
pipeline = [  
  {  
    '$match': {  
      'cntr': cntr,  
      'month': month,  
      'year': year  
    },  
    '$group': {  
      '_id': {'metro': '$metro', 'city': '$city'},  
      'averagePowerUsage': {  
        '$avg': {  
          '$divide': ['$powerUsage', '$custCnt']  
        }  
      }  
    }  
  },  
  {  
    '$sort': {'averagePowerUsage': -1},  
    '$limit': 5  
  }  
]
```

```
{  
  '$project': {  
    '_id': 0,  
    'name': { '$concat': ['$_id.metro', ' ', '$_id.city'] },  
    'averagePowerUsage': 1  
  },  
  '$sort': {'averagePowerUsage': -1},  
  '$limit': 5  
}
```

쿼리

해당 도시 계약종별 사용량

```
pipeline = [  
  {  
    '$match': {  
      'metro': metro,  
      'city': city,  
      'month': month,  
      'year': year  
    }  
  },  
  {  
    '$group': {  
      '_id': {'cntr': '$cntr'},  
      'averagePowerUsage': {  
        '$avg': {  
          '$divide': ['$powerUsage', '$custCnt']  
        }  
      }  
    }  
  }  
],
```


```
{  
  '$project': {  
    '_id': 0,  
    'name': '$_id.cntr',  
    'averagePowerUsage': 1  
  }  
}  
]
```

쿼리

해당 지역 평균 전기요금, 예상 요금

```
pipeline = [  
    {  
        "$match": {  
            "metro": metro,  
            "city": city,  
            "cntr": cntr,  
            "month": month,  
            "year": year  
        }  
    },  
    {  
        "$addFields": {  
            "billPerCust": {  
                "$divide": ["$bill", "$custCnt"]  
            }  
        }  
    }  
],
```

```
    },  
    {  
        "$group": {  
            "_id": None,  
            "averageBillPerCust": {  
                "$avg": "$billPerCust"  
            },  
            "unitCost": {  
                "$avg": "$unitCost"  
            }  
        }  
    },  
    {  
        "$project": {  
            "_id": 0,  
            "averageBillPerCust": 1,  
            "unitCost": 1  
        }  
    }  
]
```



```
my_cost = int(result[0]["unitCost"])*int(my_power_usage)
```

인덱싱

```
"$match": {  
  "metro": metro,  
  "city": city,  
  "cntr": cntr,  
  "month": month,  
  "year": year  
},
```

```
'$match': {  
  'metro': metro,  
  'city' : city,  
  'month': month,  
  'year': year  
}
```

```
{  
  '$match': {  
    'city': {'$in': nearest_city_list},  
    'metro': metro,  
    'cntr': cntr,  
    'year': year,  
    'month': month  
  },  
},
```



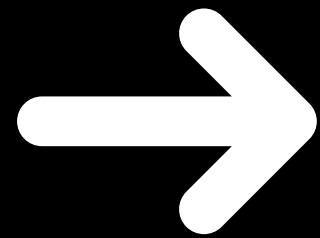
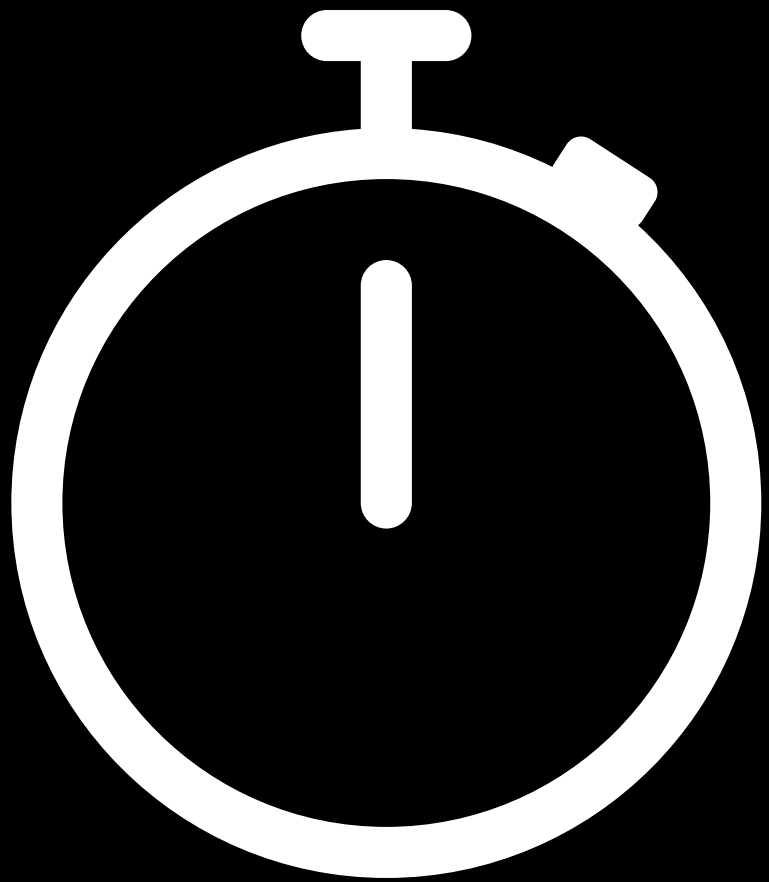
쿼리 전반적으로
거의 모든 필드가 참조됨

인덱싱

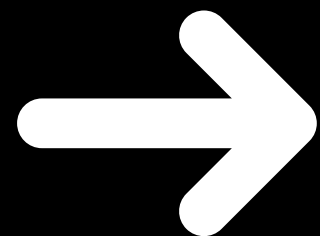
```
db.power_usage_contract.createIndex( {  
  city: 1,  
  metro: 1,  
  cntr: 1,  
  year: 1,  
  month: 1  
})
```

```
db.power_usage_contract.getIndexes()  
[  
  { v: 2, key: { _id: 1 }, name: '_id_' },  
  {  
    v: 2,  
    key: { city: 1, metro: 1, cntr: 1, year: 1, month: 1 },  
    name: 'city_1_metro_1_cntr_1_year_1_month_1'  
  }  
]
```


인덱싱 Performance



```
start_time = time.time()  
document = list(self.collection.aggregate(pipeline))  
end_time = time.time() # Get the end time  
  
execution_time = end_time - start_time # Calculate the execution time  
function_name = self.get_power_by_cntr.__name__  
print(f"{function_name} execution time: {execution_time} seconds")
```



```
start_time = time.time()  
document = list(self.collection.aggregate(pipeline))[0]  
end_time = time.time() # Get the end time  
  
execution_time = end_time - start_time # Calculate the execution time  
function_name = self.get_average_power_usage.__name__  
print(f"{function_name} execution time: {execution_time} seconds")
```


인덱싱

Performance

인덱싱 전

```
INFO: 172.30.0.2:24886 POST /api/power_usage HTTP/1.1 200 OK
get_average_power_usage execution time: 0.15114402770996094 seconds
get_average_power_usage execution time: 0.07651638984680176 seconds
get_neighbor_city_power_usage_list execution time: 0.08293724060058594 seconds
get_top_city_power_usage_list execution time: 0.09561681747436523 seconds
get_power_by_cntr execution time: 0.08744072914123535 seconds
get_my_city_cost execution time: 0.08675003051757812 seconds
INFO: 172.30.0.2:24886 "POST /api/power_usage HTTP/1.1" 200 OK
```

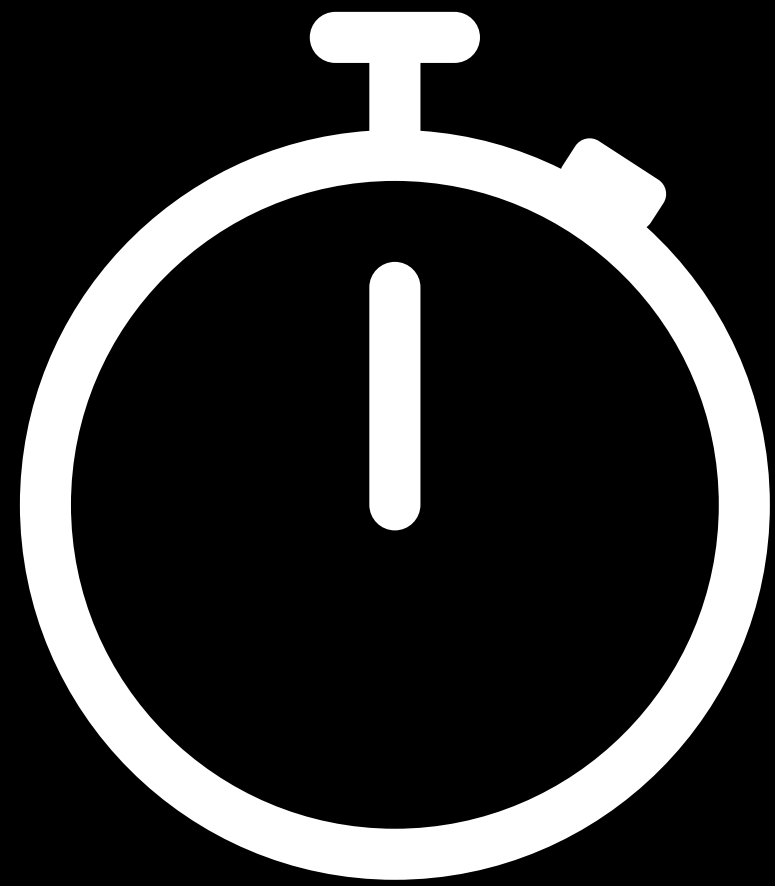
평균 : 0.11787

인덱싱 후

```
get_average_power_usage execution time: 0.00787353515625 seconds
get_average_power_usage execution time: 0.0009279251098632812 seconds
get_neighbor_city_power_usage_list execution time: 0.0013885498046875 seconds
get_top_city_power_usage_list execution time: 0.10538840293884277 seconds
get_power_by_cntr execution time: 0.0007295608520507812 seconds
get_my_city_cost execution time: 0.00035691261291503906 seconds
```

평균 : 0.02884

인덱싱 Performance



0.11787



0.02884

평균적으로 약 **75.52%** 단축

시연

Web Application



시연

<http://34.125.226.164:3000/>

Q & A

<https://github.com/power-saver>