DB2 Project Report - "Cities in Italy"

In this project, I analyzed the cities in Italy utilizing the data extracted from OpenStreetMap. Instead of data analysis, I focused more on learning **how to query the data I want from MongoDB using numerous MongoDB operators** as the main goal of "Database 2" class corresponds to it. In the following contents below, I described the process for data analysis and queries to get the information.

1. Data Collection

The data is collected from Overpass Turbo which provides the user-friendly data extraction wizard using OpenStreetMap API. I typed *place=city in Italy* in the wizard and it automatically changed to the following query. And then, I downloaded the result as a geojson format. (`city.geojson`)

```
/*
This has been generated by the overpass-turbo wizard.
The original search was:
“place=city in italy”
*/
[out:json][timeout:25];
// fetch area “italy” to search in
{{geocodeArea:italy}}->.searchArea;
// gather results
(
  // query part for: “place=city”
  node["place"="city"](area.searchArea);
  way["place"="city"](area.searchArea);
  relation["place"="city"](area.searchArea);
);
// print results
out body;
>;
out skel qt;
```

2. Import Data to MongoDB

Since the dow nloaded geojson file w as structured in only a single document, I needed to parse each node to a document to query the data w ith ease. I forked the GeoJSON to MongoDB parser from GitHub (Developer: Rick Ciesla) and executed it. This program also imports the data to MongoDB. You can check the code in `geojson-mongo-import.py` in the project folder.

3. Convert Data Type

Since `heritage` , `ele` , `postal_code` , `rank` , `population` , `admin_level` and `capital` features are stored in the database as a string type even though they are numerical data, I converted the data type to Int64. I used the follow ing queries to convert the data in MongoDB Shell.

```
# change the feature data type to numerical
db.city.find().forEach(function(data) {
    db.city.update({
        "_id": data._id,
        "properties.features_name": data.properties.features_name
    }, {
        "$set": {
            "properties.features_name": NumberInt(data.properties.features_name)
        }
    });
})
```

4. Explore the data

For analyzing the data better, good understanding of features is essential. Before executing queries, I explored the OpenStreetMap w iki so that I can understand w hat each feature exactly means.

- `admin_level` : administrative level of the city w ithin a government hierarchy. A low er level means higher in the hierarchy.
- `capital` : this property is used to tag the capital of a country or administrative divisions w ithin countries.
  - 0 : Capital city of a country
  - 4 : Capital city of an administrative region
  - 6 : Capital city of a province
- `ele` : elevation (height above sea level) of a point, in metres.
- `rank` : classification of cities
  - 0 : global city, high international importance
  - 10 : urban agglomeration
  - 20 : city w ith it's ow n metropolitan area
  - 30 : relatively small city
- `heritage` : if the value is 1, the city is registered as a heritage by World Heritage Centre.

4. Queries

In this chapter, I tried to use as many MongoDB keyw ords and operators as possible to query the interesting information about cities in Italy.

1. Field selection

```
# 1. Let's check if there is a city 'Como'!
como_city = db.city.find({"properties.name": "Como"})
```

```
# RESULT
{'_id': ObjectId('5a988542a7b7302b7c5203de'),
 'geometry': {'coordinates': [9.0863001, 45.8106992], 'type': 'Point'},
 'id': 'node/4891496721',
 'properties': {'@id': 'node/4891496721',
                'admin_level': '8',
                'capital': '6',
                'gfoss_id': '1550',
                'is_in': 'Como, Lombardia, Italy',
                'is_in:continent': 'Europe',
                'name': 'Como',
                'name:el': 'Κόμο',
                'name:fr': 'Côme',
                'name:he': 'קומו',
                'name:it': 'Como',
                'name:lmo': 'Comm',
                'name:lt': 'Komas',
                'name:ru': 'Комо',
                'name:uk': 'Комо',
                'name:zh': '科莫',
                'place': 'city',
                'population': '83422',
                'postal_code': '22100',
                'ref:ISTAT': '013075',
                'source': 'geodati.gfoss.it',
                'wikidata': 'Q1308',
                'wikipedia': 'it:Como'},
 'type': 'Feature'}
```

2. Multiple field queries

```
# 2. Are there cities that have same admin_level and capital with Como?
cities = db.city.find({"properties.admin_level": 8, "properties.capital": 6}
```

```
# RESULT
too many documents queried...
```

3. Projection queries

```
# 3. Can we make the result looks simpler?
query = {"properties.admin_level": 8, "properties.capital": 6}
projection = {"_id": 0, "properties.name": 1}
cities = db.city.find(query, projection)
```

```
# RESULT
{'properties': {'name': 'Brescia'}}
{'properties': {'name': 'Cremona'}}
{'properties': {'name': 'Lecco'}}
{'properties': {'name': 'Lodi'}}
{'properties': {'name': 'Mantova'}}
{'properties': {'name': 'Pavia'}}
{'properties': {'name': 'Sondrio'}}
{'properties': {'name': 'Varese'}}
{'properties': {'name': 'Monza'}}
{'properties': {'name': 'La Spezia'}}
{'properties': {'name': "Reggio nell'Emilia"}}
{'properties': {'name': 'Pescara'}}
{'properties': {'name': 'Bergamo'}}
{'properties': {'name': 'Como'}}
```

4. Count function

```
# 4. How many cities are in the database?
num_cities = db.city.find().count()
```

```
# RESULT
The amount of total Italian cities is 108
```

5. Range queries

```
# 5-1. Which cities has the postal code starts with 2?
query = {"properties.postal_code":{"$gte": 20000, "$lt": 30000}}
projection = {"_id": 0, "properties.name": 1, "properties.postal_code": 2}
cities = db.city.find(query, projection)
```

```
# RESULT
{'properties': {'name': 'Milano', 'postal_code': 20100}}
{'properties': {'name': 'Cremona', 'postal_code': 26100}}
{'properties': {'name': 'Como', 'postal_code': 22100}}
```

```
# 5-2. Which cities start with A?
```

```
query = {"properties.name":{"$gte": "A", "$lt": "B"}}
projection = {"_id": 0, "properties.name": 1}
cities = db.city.find(query, projection)
```

```
# RESULT
{'properties': {'name': 'Arezzo'}}
{'properties': {'name': 'Alessandria'}}
{'properties': {'name': 'Asti'}}
{'properties': {'name': 'Agrigento'}}
{'properties': {'name': 'Andria'}}
{'properties': {'name': 'Avellino'}}
{'properties': {'name': 'Ancona'}}
{'properties': {'name': 'Aosta'}}
{'properties': {'name': 'Aristanis/Oristano'}}
```

6. Exists operator

```
# 6. Which cities are registered by Unesco Heritage Centre
query = {"properties.heritage":{"$exists": 1}}
projection = {"_id": 0, "properties.name": 1}
cities = db.city.find(query, projection)
```

```
# RESULT
{'properties': {'name': 'Firenze'}}
{'properties': {'name': 'Siena'}}
{'properties': {'name': 'Mantova'}}
{'properties': {'name': 'Venezia'}}
{'properties': {'name': 'Verona'}}
{'properties': {'name': 'Vicenza'}}
{'properties': {'name': 'Catania'}}
{'properties': {'name': 'Ragusa'}}
{'properties': {'name': 'Ferrara'}}
{'properties': {'name': 'Napoli'}}
{'properties': {'name': 'Siracusa'}}
```

7. Aggregation w ith group, sort operators

```
# 7. What is the relation between capital and population?
cities = db.city.aggregate([
    {"$group": {"_id": "$properties.capital",
                "count": {"$sum": 1},
                "average_population": {"$avg": "$properties.population"}}},
    {"$sort": {"average_population": -1}}])
```

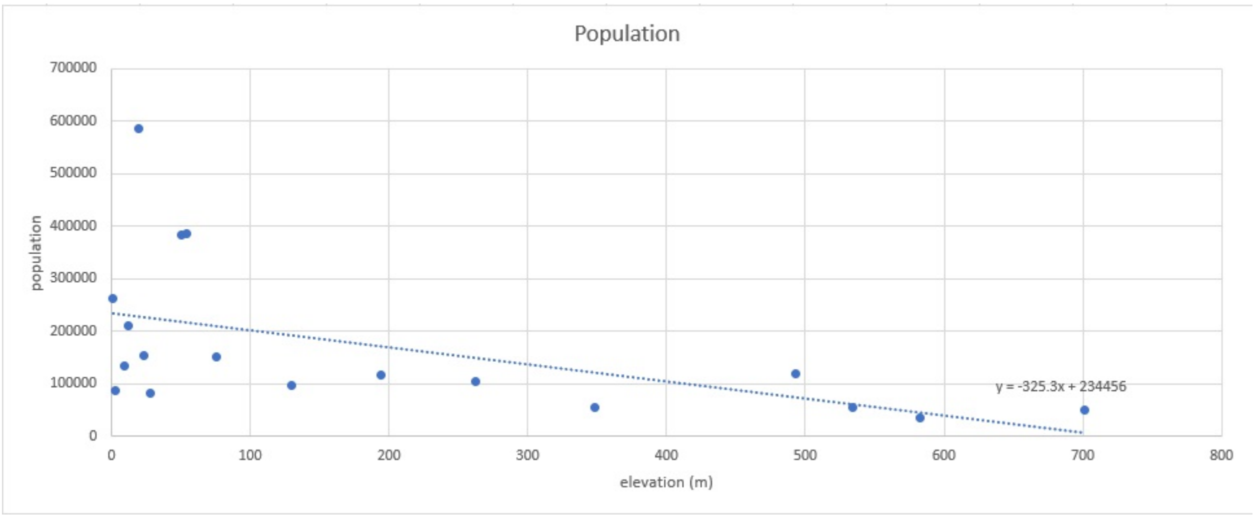
```
# RESULT
{'_id': 0, 'average_population': 2864731.0, 'count': 1}
{'_id': 4, 'average_population': 361368.5789473684, 'count': 19}
{'_id': 6, 'average_population': 99328.825, 'count': 80}
{'_id': None, 'average_population': 71113.4, 'count': 8}
```

8. Match operator

```
# 8. What is the relation between elevation and population?
{'_id': 21, 'average_population': 2864731.0, 'count': 1}
{'_id': 120, 'average_population': 1350487.0, 'count': 1}
{'_id': 17, 'average_population': 972212.0, 'count': 1}
```

```
{'_id': 239, 'average_population': 902000.0, 'count': 1}
{'_id': 20, 'average_population': 586655.0, 'count': 1}
{'_id': 54, 'average_population': 386663.0, 'count': 1}
{'_id': 50, 'average_population': 382808.0, 'count': 1}
{'_id': 1, 'average_population': 263352.0, 'count': 1}
{'_id': 12, 'average_population': 210401.0, 'count': 1}
{'_id': 23, 'average_population': 154478.0, 'count': 1}
{'_id': 76, 'average_population': 151991.0, 'count': 1}
{'_id': 9, 'average_population': 133155.0, 'count': 1}
{'_id': 493, 'average_population': 120137.0, 'count': 1}
{'_id': 194, 'average_population': 117317.0, 'count': 1}
{'_id': 262, 'average_population': 105713.0, 'count': 1}
{'_id': 130, 'average_population': 97050.0, 'count': 1}
{'_id': 3, 'average_population': 86852.0, 'count': 2}
{'_id': 28, 'average_population': 83411.0, 'count': 1}
{'_id': 534, 'average_population': 55939.0, 'count': 1}
{'_id': 348, 'average_population': 54847.0, 'count': 1}
{'_id': 701, 'average_population': 49431.0, 'count': 1}
{'_id': 583, 'average_population': 34631.0, 'count': 1}
```

```
# RESULT
{'_id': 0, 'average_population': 2864731.0, 'count': 1}
{'_id': 4, 'average_population': 361368.5789473684, 'count': 19}
{'_id': 6, 'average_population': 99328.825, 'count': 80}
{'_id': None, 'average_population': 71113.4, 'count': 8}
```



9. Project operator

```
# 9. What is the proportion of heritage cities out of each capital category?
cities = db.city.aggregate([
    {"$match": {"properties.capital": {"$exists": 1}}},
    {"$group": {"_id": "$properties.capital",
                "count": {"$sum":1},
                "num_heritage": {"$sum": "$properties.heritage"}}},
    {"$project": {"ratio": {"$divide": ["$num_heritage", "$count"]}}},
    {"$sort": {"_id": 1}}])
```

```
# RESULT
{'_id': 0, 'ratio': 0.0}
{'_id': 4, 'ratio': 0.15789473684210525}
{'_id': 6, 'ratio': 0.1}
```

10. Group accumulation operators

```
# 10-1. How many different ranks does each capital categories have?
cities = db.city.aggregate([
    {"$match": {"properties.capital": {"$exists": 1}}},
    {"$group": {"_id": "$properties.capital",
                "rank_set": {
                    "$addToSet": "$properties.rank"
                }}},
    {"$sort": {"_id": 1}}])
```

```
# RESULT
{'_id': 0, 'rank_set': [0]}
{'_id': 4, 'rank_set': [1, 10, 20]}
{'_id': 6, 'rank_set': [20, 30]}
```

```
# 10-2. How many different ranks does each capital categories have?
cities = db.city.aggregate([
    {"$match": {"properties.capital": {"$exists": 1}}},
    {"$group": {"_id": "$properties.capital",
                "rank_set": {
                    "$addToSet": "$properties.rank"
                }}},
    {"$unwind": "$rank_set"},
    {"$sort": {"_id": 1}}])
```

```
# RESULT
{'_id': 0, 'rank_set': 0}
{'_id': 4, 'rank_set': 1}
{'_id': 4, 'rank_set': 10}
{'_id': 4, 'rank_set': 20}
{'_id': 6, 'rank_set': 20}
{'_id': 6, 'rank_set': 30}
```

```
# 10-3. How many different ranks does each capital categories have?
cities = db.city.aggregate([
    {"$match": {"properties.capital": {"$exists": 1}}},
    {"$group": {"_id": "$properties.capital",
                "rank_set": {
                    "$addToSet": "$properties.rank"
                }}},
    {"$unwind": "$rank_set"},
    {"$group": {"_id": "$_id", "count": {"$sum": 1}}},
    {"$sort": {"_id": 1}}])
```

```
# RESULT
{'_id': 0, 'count': 1}
{'_id': 4, 'count': 3}
{'_id': 6, 'count': 2}
```