PROJEK IMPLEMENTASI SPASIAL DATABASE

IF4040 Pemodelan Data Lanjut

Disusun oleh:

Louis Caesa Kesuma	**13521069**
Alexander Jason	**13521100**
Juan Christopher Santoso	**13521116**

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

2024

Daftar Isi

Daftar Isi..... 1
Daftar Tabel..... 2
Daftar Gambar..... 3
A. Eksplorasi PostGIS..... 4
    1) Tipe Data Spasial..... 4
    2) Standar Data Spasial..... 5
B. Deskripsi Persoalan..... 7
C. Sumber data dan Dataset..... 8
    1) Deskripsi data..... 8
    2) Implementasi data preparation..... 11
    3) Ukuran Data..... 12
D. Pertanyaan-pertanyaan yang diajukan..... 13
E. Implementasi pada DBMS Relasional..... 15
F. Implementasi Query pada setiap DBMS..... 17
G. Hasil Query..... 21
    Atribut Deskriptif..... 21
    Atribut Spasial..... 22
    Atribut Deskriptif dan Spasial..... 25
    Interaktif..... 28
    Agregasi..... 32
H. Kesimpulan dan Lesson Learned..... 35
I. Pembagian Kerja..... 36
J. Referensi..... 37
K. Lampiran..... 38

Daftar Tabel

Tabel 1.1.1.1 Tipe-tipe Data untuk Tipe Geometri..... 4
Tabel 1.1.3.1 Tipe-tipe Data untuk 3D dan M dimensi.....5
Tabel 3.1.1 Atribut Data untuk Data Countries..... 9
Tabel 3.1.2 Atribut Data untuk Data Regions.....9
Tabel 3.1.3 Atribut Data untuk Data Districts..... 9
Tabel 3.1.4 Atribut Data untuk Data Roads..... 10
Tabel 3.1.5 Atribut Data untuk Data Rails.....10
Tabel 3.1.6 Atribut Data untuk Data Earthquake..... 10
Tabel 3.3.1 Relational Diagram Dataset Negara Inggris.....12
Tabel 4.1 Pertanyaan-pertanyaan yang diajukan..... 13
Tabel 5.1 Query untuk Membuat Tabel-tabel Pada DB.....15
Tabel 6.1 Query-query Berdasarkan Pertanyaan-pertanyaan yang Diajukan..... 17
Tabel 7.1 Hasil Eksekusi Query untuk Atribut Deskriptif.....21
Tabel 7.2 Hasil Eksekusi Query untuk Atribut Spasial.....22
Tabel 7.3 Hasil Eksekusi Query untuk Atribut Deskriptif dan Spasial.....25
Tabel 7.4 Hasil Eksekusi Query untuk Query Interaktif.....28
Tabel 7.5 Hasil Eksekusi Query untuk Query Agregasi..... 32
Tabel 9.1 Pembagian Kerja.....36

Daftar Gambar

Gambar 3.1.1 Relational Diagram Dataset Negara Inggris..... 8
Gambar 3.1.2 ERD Dataset Negara Inggris..... 8

A. Eksplorasi PostGIS

PostGIS adalah *extension spatial* untuk sistem manajemen basis data PostgreSQL yang memungkinkan penyimpanan, pengelolaan, dan analisis data geografis secara efisien. Dengan PostGIS, pengguna dapat melakukan operasi geospasial kompleks seperti pengolahan data vektor, penghitungan jarak, serta analisis topologi. Teknologi ini mendukung standar Open Geospatial Consortium (OGC), memastikan kompatibilitas dan interoperabilitas dengan berbagai aplikasi GIS lainnya. PostGIS banyak digunakan dalam pengembangan aplikasi pemetaan, analisis lokasi, dan sistem informasi geografis (SIG) karena kemampuannya yang kuat dalam menangani data geografis skala besar serta integrasinya yang seamless dengan ekosistem PostgreSQL.

1) Tipe Data Spasial

Berikut adalah data spasial yang dapat disimpan dalam PostGIS:

a. Tipe geometri

Tabel 1.1.1.1 Tipe-tipe Data untuk Tipe Geometri

Tipe Data	**Definisi**
Point	Representasi lokasi tunggal di ruang koordinat.
LineString	Representasi garis yang terdiri dari serangkaian titik terhubung.
Polygon	Representasi area tertutup dengan batas berbentuk garis.
MultiPoint	Kumpulan beberapa point dalam satu entitas geometris.
MultiLineString	Kumpulan beberapa garis dalam satu entitas geometris.
MultiPolygon	Kumpulan beberapa polygon dalam satu entitas geometris.
GeometryCollection	Koleksi dari berbagai tipe geometri (Point, LineString, Polygon, dll.) dalam satu entitas.

b. Tipe geografi
Mirip dengan tipe geometri, tetapi menggunakan sistem koordinat geografis (latitude dan longitude).

c. Tipe 3D dan M dimensi

Tabel 1.1.3.1 Tipe-tipe Data untuk 3D dan M dimensi

3D	PostGIS mendukung dimensi tambahan (z-coordinate) untuk representasi data spasial seperti PointZ, LineStringZ, dan PolygonZ.
M (Measure)	Dimensi untuk nilai tambahan, misalnya panjang atau waktu, seperti PointM, LineStringM, dan PolygonM.

d. Raster
Tipe data yang digunakan untuk menyimpan data grid berbasis piksel, seperti citra satelit atau peta topografi.

e. Tipe topologi
Representasi elemen spasial yang terhubung secara logis untuk analisis jaringan spasial, seperti jaringan jalan atau sungai.

2) Standar Data Spasial

PostGIS mematuhi standar internasional untuk data spasial yang diterapkan oleh berbagai organisasi seperti OGC dan ISO. Berikut adalah standar data spasial yang didukung:

a. Simple Features for SQL (SFS)
Standar OGC untuk manipulasi data spasial menggunakan SQL. PostGIS mendukung operasi standar seperti intersection, union, buffer, dan lainnya.

b. Well-Known Text (WKT) dan Well-Known Binary (WKB)
Format untuk representasi geometri dalam teks atau biner.

c. Coordinate Reference Systems (CRS)
Mendukung berbagai CRS untuk transformasi koordinat antara sistem yang berbeda, seperti EPSG:4326 (WGS 84) dan EPSG:3857 (Web Mercator).

d. Geography Markup Language (GML)
Format XML yang sesuai dengan standar OGC untuk pertukaran data spasial.

e. Keyhole Markup Language (KML)
Format yang digunakan untuk representasi data spasial di Google Earth dan aplikasi GIS lainnya.

f. GeoJSON
Format JSON untuk representasi geometri dan properti terkait. PostGIS menyediakan dukungan penuh untuk GeoJSON, termasuk fungsi seperti ST_AsGeoJSON untuk konversi data spasial ke format JSON.

g. ISO 19125 (Spatial Schema)
   Standar ISO untuk representasi dan manipulasi geometri spasial. PostGIS mengikuti prinsip ini melalui implementasi tipe data geometrisnya.

h. Raster dan Grid Coverage
   PostGIS mendukung standar raster seperti GeoTIFF untuk penyimpanan data berbasis piksel.

i. Topologi
   Standar OGC Topology dan ISO 19107 untuk representasi hubungan spasial antar elemen geometri.

B. Deskripsi Persoalan

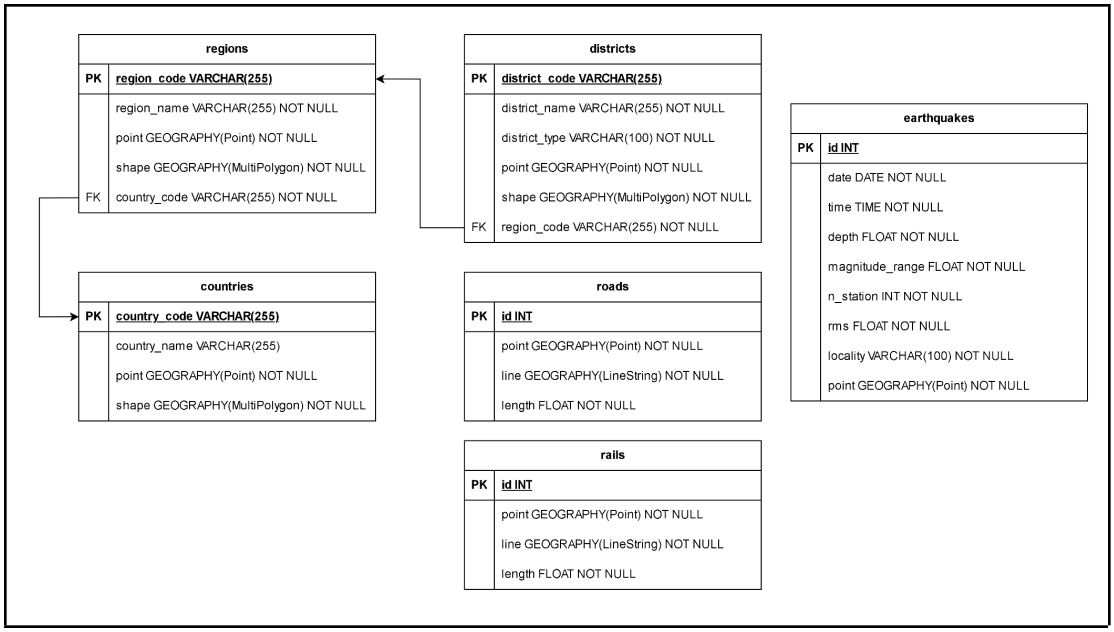
Pemerintah *United Kingdom* (Inggris) sedang berupaya dalam meningkatkan kualitas perencanaan pembangunan infrastruktur di negaranya. Namun, upaya tersebut memiliki tantangan besar berupa risiko bencana alam, khususnya gempa bumi. Walaupun gempa bumi jarang terjadi di Inggris, namun gempa bumi memiliki potensi besar untuk menimbulkan kerusakan yang besar. Untuk mengatasi hal tersebut, pemerintah memerlukan sistem *database* yang mampu menyimpan data tentang gempa bumi yang terjadi di Inggris, *region* (wilayah) terjadinya gempa, *district* (daerah) terjadinya gempa, dan *roads* (jalan) terjadinya gempa.

Database tersebut diperlukan untuk mendukung analisis mendalam mengenai pola dan dampak gempa bumi di Inggris. Informasi yang dihasilkan akan digunakan sebagai dasar pengambilan keputusan dalam perencanaan pembangunan infrastruktur di Inggris. Dengan adanya *database* tersebut, pemerintah dapat memutuskan daerah mana saja yang rentan menjadi korban dari gempa bumi, sehingga daerah tersebut dapat diprioritaskan untuk menjadi salah satu daerah yang memerlukan rencana mitigasi yang lebih efektif.

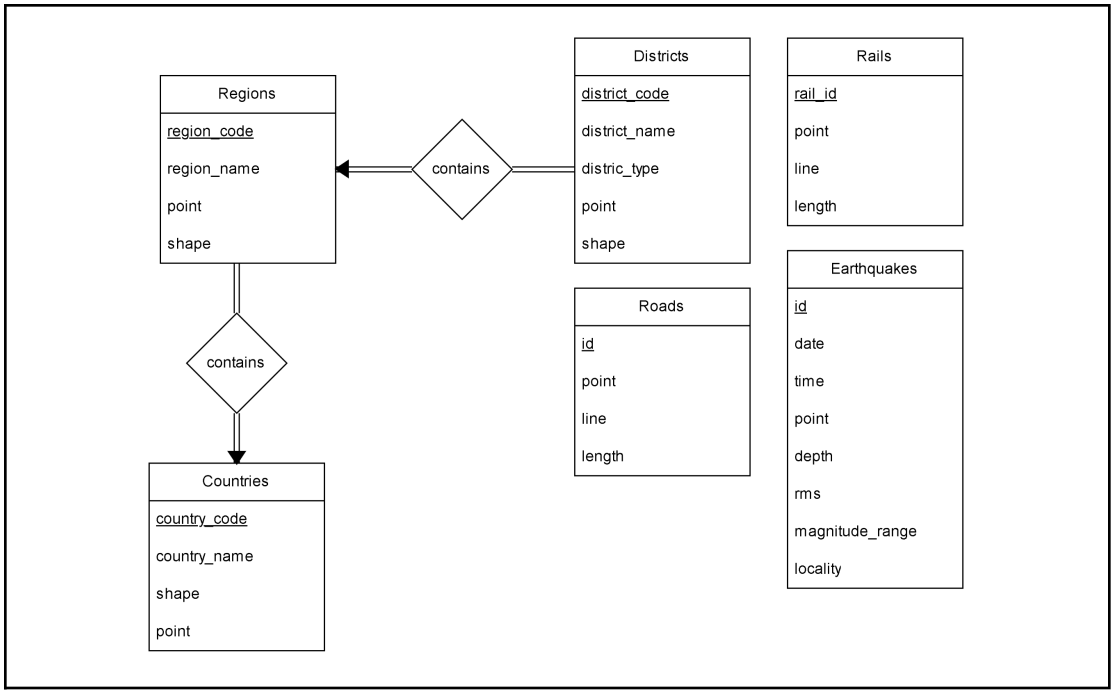
C. Sumber data dan Dataset

1) Deskripsi data

Pada projek ini, telah dilakukan *data pre-processing* berdasarkan *design* ERD sebagai berikut:



Gambar 3.1.1 Relational Diagram Dataset Negara Inggris



Gambar 3.1.2 ERD Dataset Negara Inggris

Terdapat 6 jenis dataset yang digunakan pada projek ini.

- **Data *Countries* (Tipe data *Region*),** dengan sumber berikut
  Data ini memiliki atribut sebagai berikut.

  Tabel 3.1.1 Atribut Data untuk Data *Countries*

Atribut	Deskripsi
country_code	*Identifier* unik *country*
country_name	Nama dari *country*
point	Tipe data *point* dari lokasi *country*
shape	Tipe data *region* dari lokasi *country*

- **Data *Regions* (Tipe data *Region*),** dengan sumber berikut
  Data ini memiliki atribut sebagai berikut.

  Tabel 3.1.2 Atribut Data untuk Data *Regions*

Atribut	Deskripsi
region_code	*Identifier* unik *region*
region_name	Nama dari *region*
point	Tipe data *point* dari lokasi *region*
shape	Tipe data *region* dari lokasi *region*
country_code	*Identifier* unik *country* asal *region*

- **Data *Districts* (Tipe data *Region*), berlaku sebagai sub-region, dengan sumber berikut,**
  Data ini memiliki atribut sebagai berikut.

  Tabel 3.1.3 Atribut Data untuk Data *Districts*

Atribut	Deskripsi
district_code	*Identifier* unik *district*
district_name	Nama *district*
district_type	Tipe *district*
point	Tipe data *point* dari lokasi *district*
shape	Tipe data *region* dari lokasi *district*

region_code	*Identifier* unik *region* asal *district*

- **Data *Roads* (Tipe data *Line*), dengan sumber berikut,**
  Data ini memiliki atribut sebagai berikut.

Tabel 3.1.4 Atribut Data untuk Data *Roads*

Atribut	Deskripsi
id	*Identifier* unik *road*
point	Tipe data *point* dari lokasi *road*
line	Tipe data *line* dari *road*
length	Panjang dari *road*

- **Data *Rails* (Tipe data *Line*), dengan sumber berikut,**
  Data ini memiliki atribut sebagai berikut.

Tabel 3.1.5 Atribut Data untuk Data *Rails*

Atribut	Deskripsi
id	*Identifier* unik *rail*
point	Tipe data *point* dari lokasi *rail*
line	Tipe data *line* dari *rail*
length	Panjang dari *rail*

- **Data *Earthquake* (Tipe data Point), dengan sumber berikut.**
  Data ini memiliki atribut sebagai berikut.

Tabel 3.1.6 Atribut Data untuk Data *Earthquake*

Atribut	Deskripsi
id	*Identifier* unik *earthquake*
date	Tanggal terjadinya gempa bumi
time	Waktu terjadinya gempa bumi
depth	Kedalaman gempa bumi
magnitude_range	Jangkauan area gempa bumi

n_station	Banyaknya stasiun yang mendeteksi gempa bumi
rms	*Root mean square error* titik gempa
locality	Atribut deskriptif lokai gempa bumi
point	Titik lokasi terjadinya gempa bumi

2) Implementasi *data preparation*

Data terlebih dahulu dilakukan pemrosesan data sebelum dimasukkan ke dalam DBMS. Berikut adalah langkah-langkah yang dilakukan dalam melakukan persiapan data atau pra-pemrosesan data dengan cuplikan kode dalam memproses *dataset countries*:

a) Melakukan *import* dari bentuk *.csv* ke bentuk *dataframe*.

```
country_gdf = gpd.read_file(os.path.join(DATA_DIR, 'country.csv'))
```

b) Melakukan seleksi terhadap atribut pada *dataset* dengan menghapus atribut yang tidak digunakan.

```
# Drop unnecessary columns
preserve_cols = ['Geo Point', 'Geo Shape', 'Official Code Country',
'Official Name Country']
cols = country_gdf.columns.tolist()

for col in preserve_cols:
  cols.remove(col)

country_gdf = country_gdf.drop(cols, axis=1)
```

c) Melakukan *rename* pada atribut setiap *dataset*.

```
# Rename cols
country_gdf.rename(columns={"Geo Point": "point", "Geo Shape" :
"shape", "Official Code Country" : "country_code", "Official Name
Country" : "country_name"}, inplace = True)
country_gdf.head()
```

d) Memastikan format *point* merupakan *lon-lat* (*longitude-latitude*)

```
for index, row in country_gdf.iterrows():
  lat_lon = row['point'].split(",")
  lat = lat_lon[0]
  lon = lat_lon[1]
```

```
  country_gdf.at[index, "point"] = f"{lon.strip()}, {lat.strip()}"

country_gdf.head()
```

e) Mengubah tipe data setiap *atribut* ke tipe data yang sesuai

```
# Rename cols
country_gdf['country_code'] =
country_gdf['country_code'].astype('string')
country_gdf['country_name'] =
country_gdf['country_name'].astype('string')

country_gdf.dtypes
```

f) Menyimpan hasil pra-pemrosesan data ke dalam bentuk .csv

```
country_gdf.to_csv(os.path.join(PREPROCESSED_DATA_DIR, 'country.csv'),
index=False)
```

3) Ukuran Data

Berikut adalah ukuran data untuk masing-masing *dataset* yang digunakan setelah dilakukan pra-pemrosesan:

Tabel 3.3.1 Relational Diagram Dataset Negara Inggris

No	*Dataset*	Ukuran *Storage*	Jumlah Baris	Jumlah Kolom
1	Countries	1,45 MB	4	4
2	Regions	1,16 MB	12	5
3	Districts	6,61 MB	374	6
4	Roads	12,22 MB	41460	4
5	Rails	2,43 MB	8477	4
6	Earthquakes	0,20 MB	3003	9

D. Pertanyaan-pertanyaan yang diajukan

Berikut adalah pertanyaan-pertanyaan yang kami ajukan terkait *dataset* yang digunakan:

Tabel 4.1 Pertanyaan-pertanyaan yang diajukan

Atribut deskriptif	Berapa jumlah gempa yang terjadi setiap tahun?
	Apa rentang magnitude yang paling sering terjadi, dan berapa jumlahnya?
	Berapa kedalaman rata-rata dari gempa yang terjadi di seluruh distrik?
Atribut spasial	Di mana lokasi gempa yang memiliki rentang magnitude tertentu?
	Tampilkan lokasi semua distrik yang memiliki nilai *error* gempa yang besar.
	Di wilayah geografis region tertentu, di mana saja lokasi gempa yang pernah terjadi?
Atribut deskriptif dan spasial	Apa lokasi dan kedalaman dari gempa dengan rentang magnitude tertinggi yang terjadi pada tahun tertentu?
	Tampilkan nama distrik, lokasi distrik, dan lokasi gempa dari distrik yang pernah terjadi gempa dengan kedalaman tertentu.
	Dimana saja lokasi rel kereta yang terpengaruh oleh gempa dengan rentang magnitude di atas nilai tertentu?
Interaktif	Apa detail gempa tersebut (rentang magnitude, kedalaman, waktu) jika pengguna menunjuk sebuah lokasi gempa tertentu?
	Nama distrik dan jalan yang terdampak jika pengguna menunjuk sebuah lokasi gempa tertentu?
	Apa lokasi dan rentang magnitude dari gempa yang terjadi di dalam suatu distrik jika pengguna menunjuk sebuah distrik tersebut di peta?

Agregasi	Tentukan cakupan wilayah dari semua distrik yang terdampak oleh gempa yang ditangkap oleh banyak stasiun.
	Tentukan centroid dari semua lokasi gempa yang terjadi pada tahun tertentu.
	Tentukan area geografis gabungan (spatial union) dari semua gempa yang terjadi di dalam wilayah distrik tertentu.

E. Implementasi pada DBMS Relasional

Berikut adalah query untuk membuat database.

```
CREATE DATABASE earthquake
```

Berikut adalah query untuk membuat PostGIS *extension*.

```
CREATE EXTENSION IF NOT EXISTS postgis
```

Berikut adalah query untuk membuat tabel-tabel pada DB.

Tabel 5.1 Query untuk Membuat Tabel-tabel Pada DB

Countries

```
CREATE TABLE IF NOT EXISTS countries (
  country_code VARCHAR(255) PRIMARY KEY,
  country_name VARCHAR(255) NOT NULL,
  point GEOGRAPHY(Point) NOT NULL,
  shape GEOGRAPHY(MultiPolygon) NOT NULL
)
```

Regions

```
CREATE TABLE IF NOT EXISTS regions (
  region_code VARCHAR(255) PRIMARY KEY,
  region_name VARCHAR(255) NOT NULL,
  point GEOGRAPHY(Point) NOT NULL,
  shape GEOGRAPHY(MultiPolygon) NOT NULL,
  country_code VARCHAR(255) NOT NULL,
  FOREIGN KEY (country_code) REFERENCES countries(country_code) ON
UPDATE CASCADE ON DELETE CASCADE
)
```

Districts

```
CREATE TABLE IF NOT EXISTS districts (
  district_code VARCHAR(255) PRIMARY KEY,
  district_name VARCHAR(255) NOT NULL,
  district_type VARCHAR(100) NOT NULL,
  point GEOGRAPHY(Point) NOT NULL,
  shape GEOGRAPHY NOT NULL,
  region_code VARCHAR(255) NOT NULL,
  FOREIGN KEY (region_code) REFERENCES regions(region_code) ON UPDATE
CASCADE ON DELETE CASCADE,
  CONSTRAINT check_polygon_multipolygon CHECK (
      GeometryType(shape) IN ('POLYGON', 'MULTIPOLYGON')
  )
)
```

Roads

```
CREATE TABLE IF NOT EXISTS roads (
```

```
  id INT PRIMARY KEY,
  point GEOGRAPHY(Point) NOT NULL,
  line GEOGRAPHY(LineString) NOT NULL,
  length FLOAT NOT NULL
)
```

Rails

```
CREATE TABLE IF NOT EXISTS rails (
  id INT PRIMARY KEY,
  point GEOGRAPHY(Point) NOT NULL,
  line GEOGRAPHY(LineString) NOT NULL,
  length FLOAT NOT NULL
)
```

Earthquakes

```
CREATE TABLE IF NOT EXISTS earthquakes (
  id INT PRIMARY KEY,
  date DATE NOT NULL,
  time TIME NOT NULL,
  depth FLOAT NOT NULL,
  magnitude_range FLOAT NOT NULL,
  n_station INT NOT NULL,
  rms FLOAT NOT NULL,
  locality VARCHAR(100) NOT NULL,
  point GEOGRAPHY(Point) NOT NULL
)
```

F. Implementasi Query pada setiap DBMS

Berikut merupakan implementasi *query* dari pertanyaan-pertanyaan yang telah didefinisikan:

Tabel 6.1 Query-query Berdasarkan Pertanyaan-pertanyaan yang Diajukan

Atribut deskriptif	Query 1	`SELECT` `  EXTRACT(YEAR FROM date) AS year,` `  COUNT(id) AS total_earthquakes` `FROM` `  Earthquakes` `GROUP BY` `  EXTRACT(YEAR FROM date)` `ORDER BY` `  year;`
	Query 2	`SELECT` `  magnitude_range, COUNT(id) AS total_earthquakes` `FROM` `  Earthquakes` `GROUP BY magnitude_range` `ORDER BY total_earthquakes` `DESC LIMIT 1;`
	Query 3	`SELECT` `  AVG(depth) AS average_depth` `FROM` `  Earthquakes;`
Atribut spasial	Query 1	`SELECT` `  id,` `  ST_AsText(e.point) AS location,` `  magnitude_range` `FROM` `  Earthquakes as e` `WHERE` `  magnitude_range BETWEEN 4 AND 5;`
	Query 2	`SELECT` `  e.id,` `  ST_AsText(e.point) AS location,` `  d.district_name` `FROM` `  Earthquakes e` `JOIN` `  Districts d` `ON` `  ST_Contains(d.shape::geometry,` `e.point::geometry)` `WHERE` `  e.rms > 1;`
	Query 3	`SELECT` `  e.id AS earthquake_id,` `  ST_AsText(e.point) AS location`

		FROM Earthquakes e JOIN Regions r ON ST_Contains(r.shape::geometry, e.point::geometry) WHERE r.region_name = 'South East';
Atribut deskriptif dan spasial	Query 1	SELECT id, ST_AsText(point) AS location, depth, magnitude_range FROM Earthquakes WHERE EXTRACT(YEAR FROM date) = 2023 ORDER BY magnitude_range DESC LIMIT 1;
	Query 2	SELECT DISTINCT d.district_name, ST_AsText(e.point) AS point, ST_AsText(d.shape) AS district_location FROM Earthquakes e JOIN Districts d ON ST_Contains(d.shape::geometry, e.point::geometry) WHERE e.depth > 30;
	Query 3	SELECT e.id, r.id AS road_id, ST_AsText(e.point) AS earthquake_location, ST_AsText(r.line) AS rail_line FROM Earthquakes e JOIN Rails r ON ST_DWithin(r.line::geometry, e.point::geometry, 0.05) WHERE e.magnitude_range > 2;
Interaktif	Query 1	SELECT id, depth, magnitude_range,

		date, time, ST_AsText(point) as earthquake_point FROM Earthquakes WHERE ST_DWithin(point::geometry, ST_SetSRID(ST_MakePoint('-0.2', '53'), 4326), 0.32);
	Query 2	SELECT d.district_name, r.id AS road_id, ST_AsText(r.line) AS road_line FROM Districts d JOIN Roads r ON ST_Intersects(d.shape::geometry, r.line::geometry) WHERE ST_Contains(d.shape::geometry, ST_SetSRID(ST_MakePoint('0.4', '51'), 4326));
	Query 3	SELECT e.id, ST_AsText(e.point) AS location, e.magnitude_range FROM Earthquakes e JOIN Districts d ON ST_Contains(d.shape::geometry, e.point::geometry) WHERE ST_Contains(d.shape::geometry, ST_SetSRID(ST_MakePoint('-0.3', '53'), 4326));
Agregasi	Query 1	SELECT ST_AsText(ST_UNION(d.shape::geometry)) FROM Districts d JOIN Earthquakes e ON ST_Contains(d.shape::geometry, e.point::geometry) WHERE e.n_station > 35;
	Query 2	SELECT ST_AsText(ST_Centroid(ST_Collect(point::geometry))) AS centroid_location FROM

		Earthquakes WHERE EXTRACT(YEAR FROM date) = 2023;
	Query 3	SELECT ST_AsText(ST_Union(e.point::geometry)) AS earthquake_area FROM Earthquakes e JOIN Districts d ON ST_Contains(d.shape::geometry, e.point::geometry) WHERE d.district_name = 'Conwy';

G. Hasil Query

Berikut adalah hasil *screenshot* dari pelaksanaan setiap *query* yang telah disajikan pada bagian F:

Tabel 7.1 Hasil Eksekusi Query untuk Atribut Deskriptif

Atribut Deskriptif

Query 1

```
SELECT
    EXTRACT(YEAR FROM date) AS year,
    COUNT(id) AS total_earthquakes
FROM
    Earthquakes
GROUP BY
    EXTRACT(YEAR FROM date)
ORDER BY
    year;
```

	year numeric	total_earthquakes bigint
1	2014	14
2	2015	258
3	2016	205
4	2017	209
5	2018	287
6	2019	449
7	2020	430
8	2021	320
9	2022	288
10	2023	298
11	2024	245

Query 2

```
SELECT
    magnitude_range, COUNT(id) AS total_earthquakes
FROM
    Earthquakes
GROUP BY magnitude_range
ORDER BY total_earthquakes
DESC LIMIT 1;
```

	magnitude_range double precision	total_earthquakes bigint
1	0.8	215

Query 3	SELECT AVG(depth) AS average_depth FROM Earthquakes;

	average_depth double precision
1	7.012487512487465

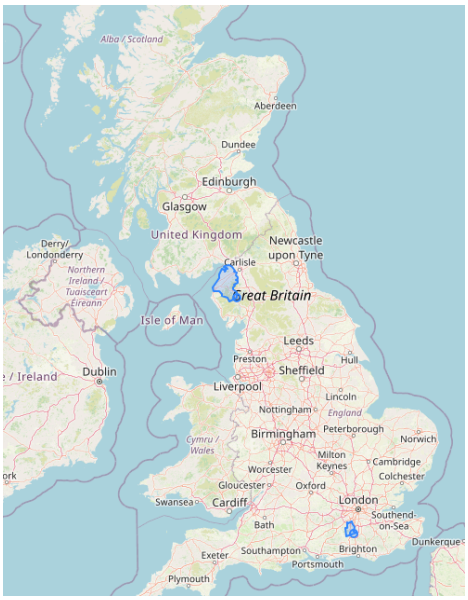
Tabel 7.2 Hasil Eksekusi Query untuk Atribut Spasial

Atribut Spasial	
Query 1	SELECT id, ST_AsText(e.point) AS location, magnitude_range FROM Earthquakes as e WHERE magnitude_range BETWEEN 4 AND 5;
Query Visualisasi	SELECT id, e.point AS location, magnitude_range FROM Earthquakes as e WHERE magnitude_range BETWEEN 4 AND 5;

	id [PK] integer	location text	magnitude_range double precision
1	103	POINT(51.304 1.438)	4.2
2	158	POINT(53.176 2.168)	4.1
3	573	POINT(58.952 1.866)	4.7
4	591	POINT(56.805 -5.888)	4
5	704	POINT(51.767 -3.833)	4.6
6	1370	POINT(57.034 1.935)	4.2
7	1467	POINT(55.679 3.002)	4
8	2380	POINT(61.804 4.129)	4.2

Query 2	SELECT e.id, ST_AsText(e.point) AS location, d.district_name FROM Earthquakes e JOIN Districts d ON ST_Contains(d.shape::geometry, e.point::geometry) WHERE e.rms > 1;
Query Visualisasi	SELECT e.id, ST_COLLECT(e.point::geometry, d.shape::geometry), d.district_name FROM Earthquakes e JOIN Districts d ON ST_Contains(d.shape::geometry, e.point::geometry) WHERE e.rms > 1;

	id integer	district_name character varying (255)	location text	st_astext text
1	721	Allerdale	POINT(-3.018 54.506)	POLYGON((-3.118792037506064 54.92
2	801	Mole Valley	POINT(-0.238 51.153)	POLYGON((-0.272258875626302 51.29



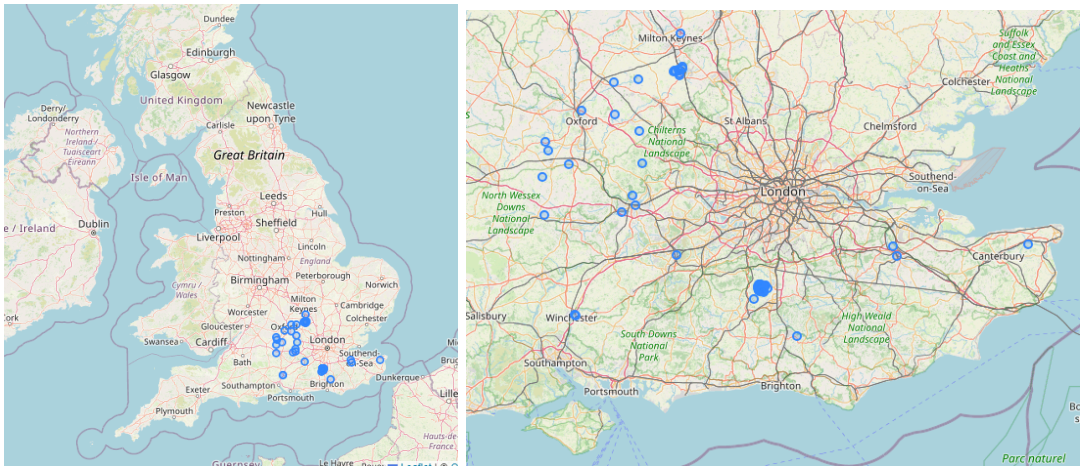
Query 3	`SELECT` `  e.id AS earthquake_id,` `  ST_AsText(e.point) AS location` `FROM` `  Earthquakes e` `JOIN` `  Regions r` `ON` `  ST_Contains(r.shape::geometry, e.point::geometry)` `WHERE` `  r.region_name = 'South East';`
Query Visualisasi	`SELECT` `  e.id AS earthquake_id,` `  ST_AsText(e.point) AS location` `FROM` `  Earthquakes e` `JOIN` `  Regions r` `ON` `  ST_Contains(r.shape::geometry, e.point::geometry)` `WHERE` `  r.region_name = 'South East';`

	earthquake_id integer	location text
1	28	POINT(-1.299 51.072)
2	31	POINT(-1.289 51.069)
3	277	POINT(0.512 51.28)
4	308	POINT(-0.935 51.718)
5	376	POINT(-1.475 51.558)
6	483	POINT(-1.446 51.648)
7	741	POINT(-0.265 51.173)
8	742	POINT(-0.261 51.157)
9	743	POINT(-0.244 51.166)
10	747	POINT(-1.328 51.6)
11	749	POINT(-0.047 50.997)
12	765	POINT(-0.234 51.163)
13	801	POINT(-0.238 51.153)
14	804	POINT(-0.264 51.178)
15	808	POINT(-0.246 51.163)
16	810	POINT(-0.233 51.171)

Total rows: 185 of 185 Query complete 00:00:00.167

	earthquake_id integer	location text
171	1642	POINT(-0.73 51.925)
172	1644	POINT(-0.74 51.924)
173	1649	POINT(-0.712 51.93)
174	1818	POINT(-0.252 51.155)
175	1842	POINT(-0.24 51.16)
176	1851	POINT(0.487 51.314)
177	2179	POINT(-0.957 51.458)
178	2180	POINT(-1.035 51.435)
179	2438	POINT(-1.469 51.424)
180	2623	POINT(-0.706 51.927)
181	2655	POINT(1.248 51.321)
182	2718	POINT(-0.705 52.056)
183	2732	POINT(-0.725 51.284)
184	2970	POINT(-1.462 51.681)
185	3001	POINT(-1.073 51.775)

Total rows: 185 of 185 Query complete 00:00:00.167



Tabel 7.3 Hasil Eksekusi Query untuk Atribut Deskriptif dan Spasial

Atribut Deskriptif dan Spasial	
Query 1	`SELECT` `  id,` `  ST_AsText(point) AS location,` `  depth,` `  magnitude_range` `FROM` `  Earthquakes` `WHERE` `  EXTRACT(YEAR FROM date) = 2023` `ORDER BY` `  magnitude_range DESC` `LIMIT 1;`
Query Visualisasi	`SELECT` `  id,` `  (point) AS location,` `  depth,` `  magnitude_range` `FROM` `  Earthquakes` `WHERE` `  EXTRACT(YEAR FROM date) = 2023` `ORDER BY` `  magnitude_range DESC` `LIMIT 1;`

	id [PK] integer	location text	depth double precision	magnitude_range double precision
1	2685	POINT(4.805 62.328)	10.2	3.9

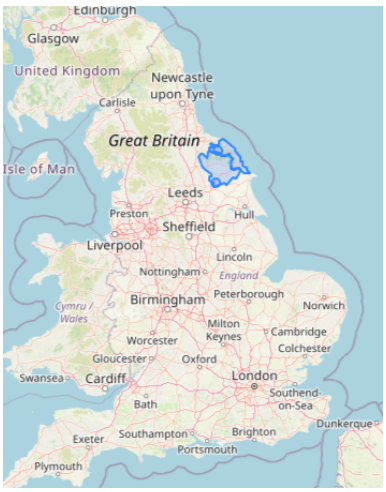
Query 2

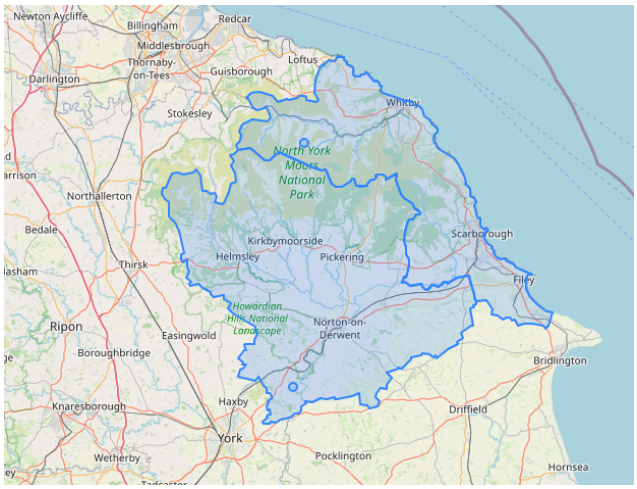
```
SELECT
    DISTINCT d.district_name,
      ST_AsText(e.point) AS point,
    ST_AsText(d.shape) AS district_location
FROM
    Earthquakes e
JOIN
    Districts d
ON
    ST_Contains(d.shape::geometry, e.point::geometry)
WHERE
    e.depth > 30;
```

Query Visualisasi

```
SELECT
    DISTINCT d.district_name,
      ST_COLLECT(e.point::geometry, d.shape::geometry)
FROM
    Earthquakes e
JOIN
    Districts d
ON
    ST_Contains(d.shape::geometry, e.point::geometry)
WHERE
    e.depth > 30;
```

	district_name character varying (255)	point text	district_location text
1	Ryedale	POINT(-0.912 54.04)	POLYGON((-0.418235895489948 54.17404511421572,-0.418646541550253 54.175532729
2	Scarborough	POINT(-0.881 54.423)	POLYGON((-1.059367627242685 54.412565619113,-1.026859274890195 54.4153314625

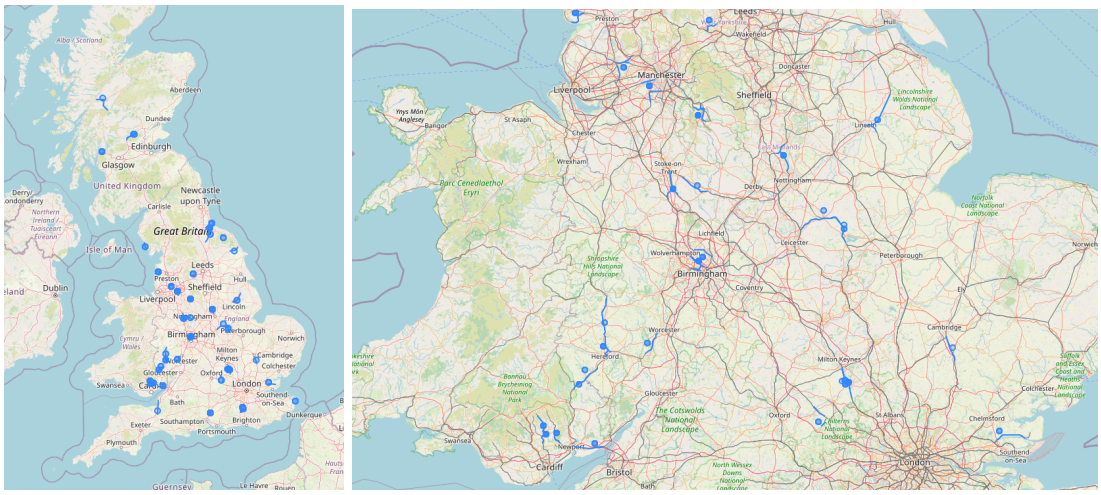




Query 3	`SELECT` `  e.id,` `  r.id AS road_id,` `  ST_AsText(e.point) AS earthquake_location,` `  ST_AsText(r.line) AS rail_line` `FROM` `  Earthquakes e` `JOIN` `  Rails r` `ON` `  ST_DWithin(r.line::geometry, e.point::geometry, 0.05)` `WHERE` `  e.magnitude_range > 2;`
Query Visualisasi	`SELECT` `  e.id,` `  r.id AS road_id,` `  ST_COLLECT(e.point::geometry, r.line::geometry)` `FROM` `  Earthquakes e` `JOIN` `  Rails r` `ON` `  ST_DWithin(r.line::geometry, e.point::geometry, 0.05)` `WHERE` `  e.magnitude_range > 2;`

	id integer	road_id integer	earthquake_location text	rail_line text
1	1107	34	POINT(-3.382 54.252)	LINESTRING(-3.38945000000021 54.28028199999982,-3.3890820000002 54.2791194
2	1534	213	POINT(-1.903 52.562)	LINESTRING(-1.894105000000195 52.60023549999982,-1.910049000000185 52.6010
3	2309	252	POINT(-2.348 53.427)	LINESTRING(-2.249664000000194 53.39903299999983,-2.254466000000207 53.3992
4	1653	257	POINT(-3.735 56.274)	LINESTRING(-3.683919000000202 56.292444999999844,-3.690379500000204 56.290
5	1633	340	POINT(-0.737 51.926)	LINESTRING(-0.735736000000202 51.99290149999984,-0.734718000000186 51.9892
6	1641	340	POINT(-0.709 51.911)	LINESTRING(-0.735736000000202 51.99290149999984,-0.734718000000186 51.9892
7	1648	340	POINT(-0.679 51.919)	LINESTRING(-0.735736000000202 51.99290149999984,-0.734718000000186 51.9892
8	1649	340	POINT(-0.712 51.93)	LINESTRING(-0.735736000000202 51.99290149999984,-0.734718000000186 51.9892

Total rows: 235 of 235 | Query complete 00:00:05.281 | Ln 14, Col 1

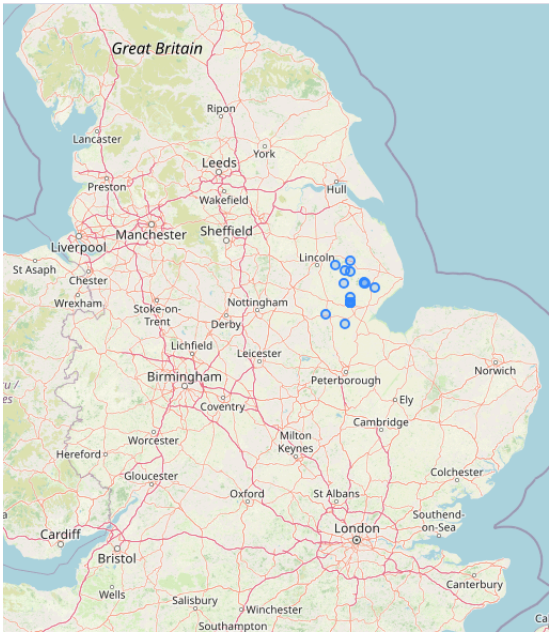


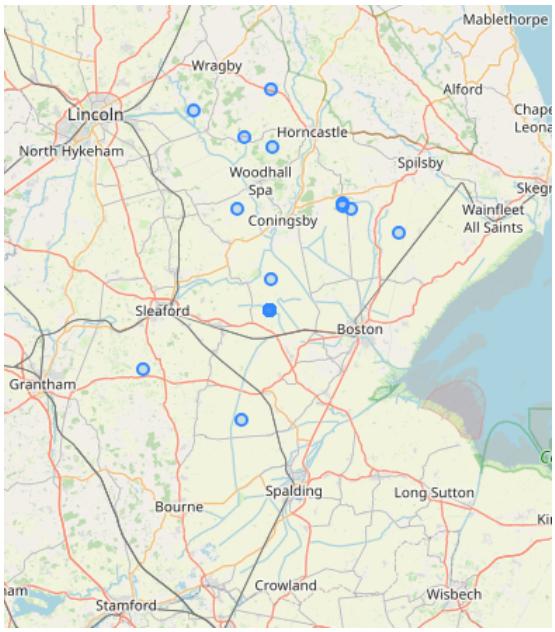
Tabel 7.4 Hasil Eksekusi Query untuk Query Interaktif

Interaktif	
Query 1	`SELECT` `  id,` `  depth,` `  magnitude_range,` `  date,` `  time,` `  ST_AsText(point) as earthquake_point` `FROM` `  Earthquakes` `WHERE` `  ST_DWithin(point::geometry,` `ST_SetSRID(ST_MakePoint('-0.2', '53'), 4326), 0.32);`
Query Visualisasi	`SELECT` `  id,` `  depth,` `  magnitude_range,` `  date,` `  time,`

```
      point as earthquake_point
FROM
    Earthquakes
WHERE
    ST_DWithin(point::geometry,
ST_SetSRID(ST_MakePoint('-0.2', '53'), 4326), 0.32);
```

	id [PK] integer	depth double precision	magnitude_range double precision	date date	time time without time zone	earthquake_point text
1	524	15.5	1.7	2017-03-04	19:54:18	POINT(-0.198 53.259)
2	601	2.4	1.4	2017-08-11	03:44:08	POINT(-0.35 53.234)
3	689	3.2	1	2018-01-09	22:43:29	POINT(0.051 53.091)
4	995	26.8	2.2	2019-01-27	23:48:19	POINT(-0.25 53.202)
5	1426	7.9	1.5	2019-12-23	23:43:56	POINT(-0.199 53.037)
6	1983	7.5	1.8	2021-03-30	06:57:49	POINT(-0.255 52.871)
7	2028	11.8	2.1	2021-06-02	14:58:51	POINT(-0.058 53.122)
8	2291	14.1	2.2	2022-05-12	03:57:24	POINT(-0.042 53.119)
9	2316	7.6	1.3	2022-06-01	18:15:29	POINT(-0.058 53.126)
10	2575	29.8	2.2	2023-03-29	18:24:28	POINT(-0.195 53.191)
11	2772	2.6	1.2	2024-01-10	18:05:09	POINT(-0.263 53.118)
12	2931	7.6	1.3	2024-08-06	21:16:37	POINT(-0.448 52.931)



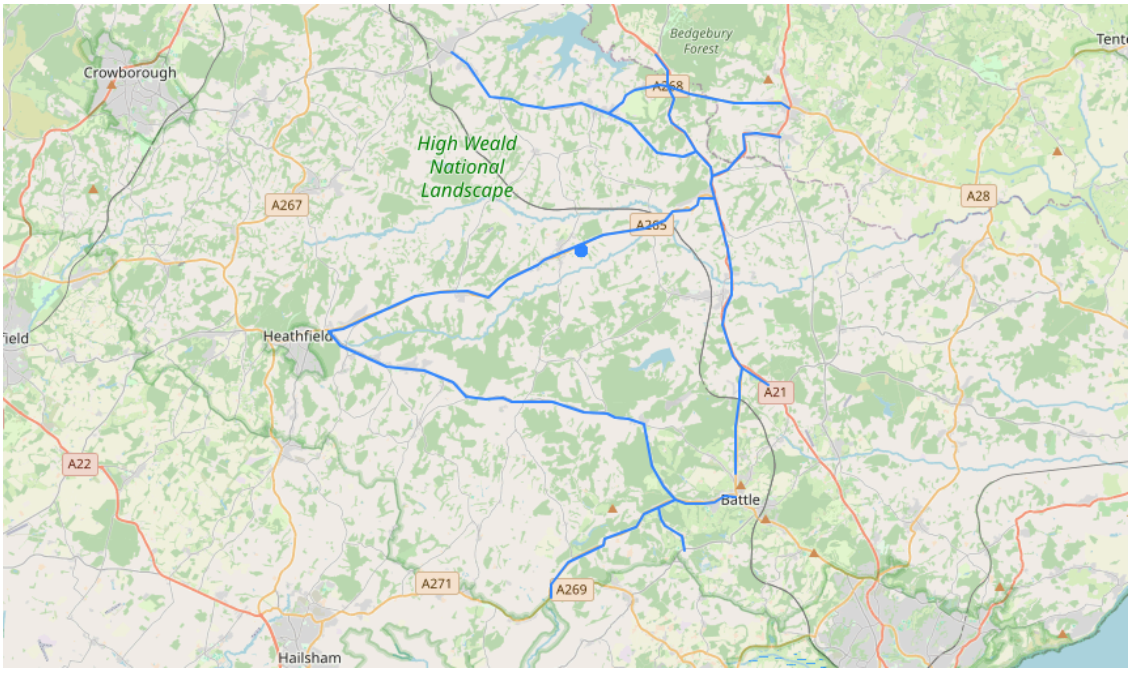


Query 2

```
SELECT
    d.district_name,
    r.id AS road_id,
      ST_AsText(r.line) AS road_line
FROM
    Districts d
JOIN
    Roads r
ON
    ST_Intersects(d.shape::geometry, r.line::geometry)
```

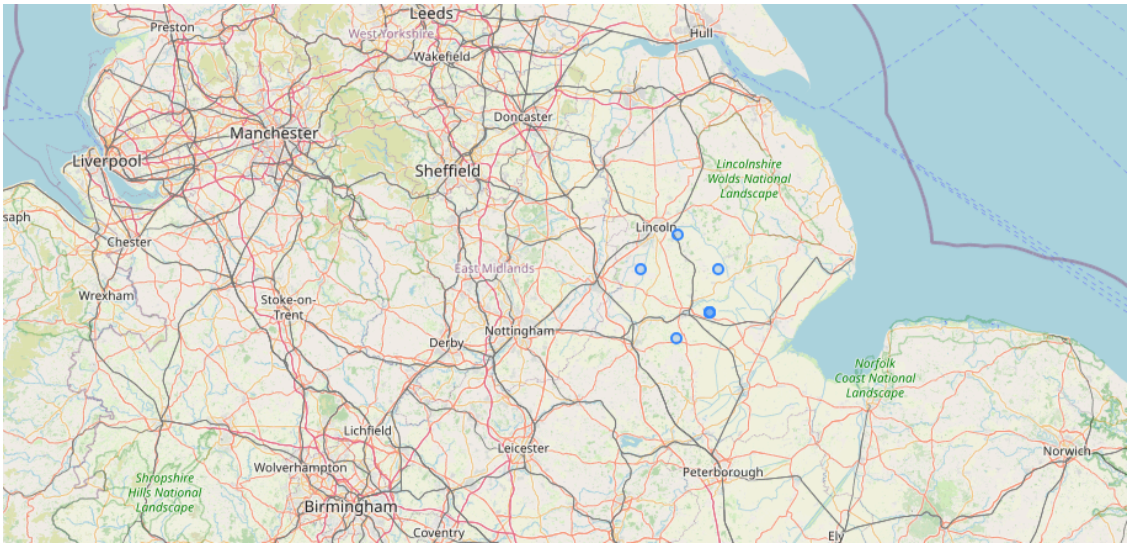
	WHERE ST_Contains(d.shape::geometry, ST_SetSRID(ST_MakePoint('0.4', '51'), 4326)) AND ST_DWithin(r.line::geometry, ST_SetSRID(ST_MakePoint('0.4', '51'), 4326), 0.1);
Query Visualisasi	SELECT d.district_name, r.id AS road_id, ST_Collect(r.line::geometry, ST_SetSRID(ST_MakePoint('0.4', '51'), 4326)) FROM Districts d JOIN Roads r ON ST_Intersects(d.shape::geometry, r.line::geometry) WHERE ST_Contains(d.shape::geometry, ST_SetSRID(ST_MakePoint('0.4', '51'), 4326)) AND ST_DWithin(r.line::geometry, ST_SetSRID(ST_MakePoint('0.4', '51'), 4326), 0.1);

	district_name character varying (255)	road_id integer	road_line text
1	Rother	1447	LINESTRING(0.761509499999789 50.96973749999984,0.773140999999811 50.97199549999985,0.779987999999804 50.97578949999985,0.783415499999791 50.975540499
2	Rother	2155	LINESTRING(0.332237499999792 51.06561499999984,0.340744999999799 51.06166749999983,0.345476499999791 51.05644599999985,0.352183499999796 51.051085499
3	Rother	2743	LINESTRING(0.612946999999792 51.00650799999855,0.617748999999804 51.00064449999844,0.625639999999805 50.99902799999983,0.635594999999796 50.998623
4	Rother	4211	LINESTRING(0.523876999999999 50.96215599999844,0.526309999999796 50.96263399999984,0.527913499999812 50.96278799999985)
5	Rother	4295	LINESTRING(0.597490499999793 50.94928899999844,0.600035999999979 50.95382849999844,0.597927999999797 50.95458599999983,0.594116999999812 50.962855499
6	Rother	4413	LINESTRING(0.445683499999802 51.05447199999983,0.448871999999795 51.04999799999846,0.446187999999808 51.04495499999845,0.447889999999802 51.042193999
7	Rother	4710	LINESTRING(0.492163499999805 50.91125049999985,0.491282499999812 50.91194849999985)
8	Rother	4798	LINESTRING(0.26777149999981 50.97316799999845,0.273794999999808 50.96845499999835,0.290667999999812 50.96398799999982,0.297946999999795 50.961416499
9	Rother	4914	LINESTRING(0.379057999999787 50.84853399999983,0.384780999999805 50.84931649999984,0.394632999999999 50.84749599999836,0.400883999999792 50.844395499
10	Rother	5111	LINESTRING(0.492163499999805 50.91125049999985,0.493052499999806 50.91054799999835,0.493622999999786 50.91022549999854,0.499256999999801 50.909700499
11	Rother	6074	LINESTRING(0.498599999999793 50.95550449999984,0.507875999999811 50.94828099999984,0.511071499999787 50.944164999999856,0.518592999999811 50.935817999
12	Rother	6212	LINESTRING(0.605781999999806 50.98778899999985,0.602989499999808 50.99161999999834,0.599790999999811 50.99304049999836,0.599579999999804 50.996912999
13	Rother	7206	LINESTRING(0.449429499999809 50.91714299999984,0.454929999999791 50.91621799999844,0.464740999999805 50.91610099999856,0.470174999999813 50.916615499
14	Rother	8017	LINESTRING(0.527913499999812 50.96278799999985,0.529316999999793 50.96292849999983)
15	Rother	8027	LINESTRING(0.597490499999793 50.94928899999844,0.605428999999788 50.94866249999834,0.608548499999813 50.94589449999983,0.614891999999799 50.944762999
16	Rother	8131	LINESTRING(0.487957999999793 50.84638149999983,0.50400399999981 50.84608449999985)



Query 3	`SELECT` `  e.id,` `  ST_AsText(e.point) AS location,` `  e.magnitude_range` `FROM` `  Earthquakes e` `JOIN` `  Districts d` `ON` `  ST_Contains(d.shape::geometry, e.point::geometry)` `WHERE` `  ST_Contains(d.shape::geometry,` `ST_SetSRID(ST_MakePoint('-0.3', '53'), 4326));`
Query Visualisasi	`SELECT` `  e.id,` `  ST_Collect(e.point::geometry,` `ST_SetSRID(ST_MakePoint('-0.3', '53'), 4326)),` `  e.magnitude_range` `FROM` `  Earthquakes e` `JOIN` `  Districts d` `ON` `  ST_Contains(d.shape::geometry, e.point::geometry)` `WHERE` `  ST_Contains(d.shape::geometry,` `ST_SetSRID(ST_MakePoint('-0.3', '53'), 4326));`

	id [PK] integer	location text	magnitude_range double precision
1	341	POINT(-0.607 53.118)	1.3
2	932	POINT(-0.444 53.209)	0.9
3	2772	POINT(-0.263 53.118)	1.2
4	2931	POINT(-0.448 52.931)	1.3

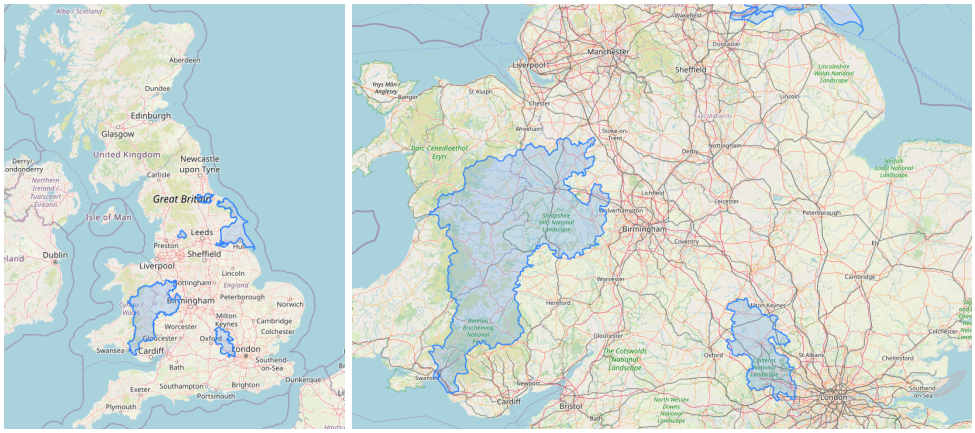


Tabel 7.5 Hasil Eksekusi Query untuk Query Agregasi

Agregasi	
Query 1	`SELECT` `  ST_AsText(ST_UNION(d.shape::geometry))` `FROM` `  Districts d` `JOIN` `  Earthquakes e` `ON` `  ST_Contains(d.shape::geometry, e.point::geometry)` `WHERE` `  e.n_station > 35;`
Query Visualisasi	`SELECT` `  ST_UNION(d.shape::geometry)` `FROM` `  Districts d` `JOIN` `  Earthquakes e` `ON` `  ST_Contains(d.shape::geometry, e.point::geometry)` `WHERE`

	`e.n_station > 35`

	st_astext text
1	MULTIPOLYGON(((-3.591286561788011 51.74896662010835,-3.581597524622919 51.73518791990243,-3.580196185506929 51.72839448586355,-3.579703824501525 51.72345373756701,-3.5



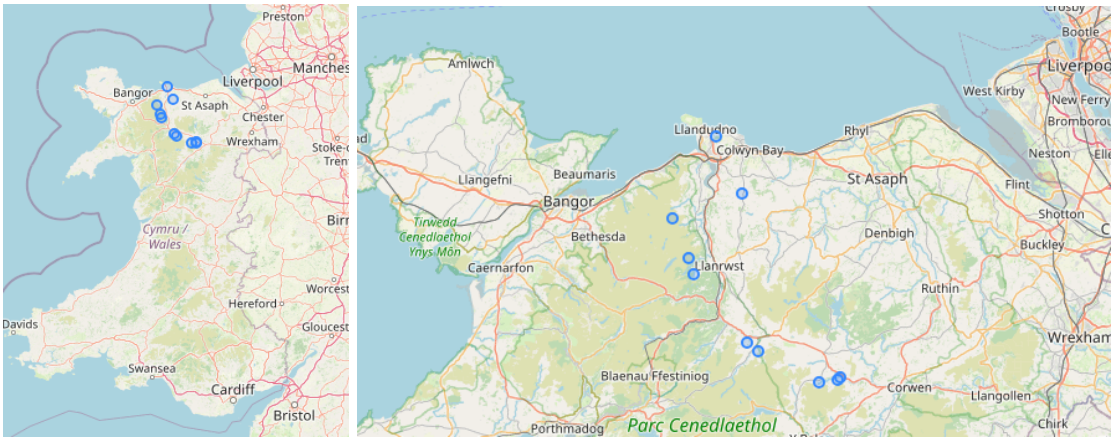
Query 2	`SELECT` `    ST_AsText(ST_Centroid(ST_Collect(point::geometry))) AS centroid_location` `FROM` `    Earthquakes` `WHERE` `    EXTRACT(YEAR FROM date) = 2023;`
Query Visualisasi	`SELECT` `    ST_Centroid(ST_Collect(point::geometry)) AS centroid_location` `FROM` `    Earthquakes` `WHERE` `    EXTRACT(YEAR FROM date) = 2023;`

	centroid_location text
1	POINT(-4.133617449664431 54.58695302013424)

Query 3	`SELECT`

	`` ST_AsText(ST_Union(e.point::geometry)) AS earthquake_area FROM Earthquakes e JOIN Districts d ON ST_Contains(d.shape::geometry, e.point::geometry) WHERE d.district_name = 'Conwy'; ```
Query Visualisasi	``` SELECT ST_Union(e.point::geometry) AS earthquake_area FROM Earthquakes e JOIN Districts d ON ST_Contains(d.shape::geometry, e.point::geometry) WHERE d.district_name = 'Conwy'; ```

	earthquake_area text
1	MULTIPOINT((-3.897 53.204),(-3.862 53.152),(-3.852 53.131),(-3.801 53.313),(-3.743 53.237),(-3.732 53.039),(-3.709 53.028),(-3.574 52.987),(-3.53



H. Kesimpulan dan *Lesson Learned*

Implementasi data *spatial* menggunakan PostGIS menjadi solusi yang efektif untuk penyimpanan, pengelolaan, dan analisis data geospasial dalam PostgreSQL. PostGIS mempermudah pemrosesan data geospasial dengan mendukung berbagai tipe data geometris serta fungsi spatial yang kompleks, seperti Point, LineString, Polygon, dan MultiPolygon. Keunggulan lainnya adalah kemudahan integrasi dengan berbagai aplikasi GIS, skalabilitas yang baik, dan dukungan terhadap proyeksi geografis.

Lesson learned yang didapat dari projek ini adalah sebagai berikut:

- Desain basis data dan pemilihan tipe data geometris yang tepat sangat berpengaruh terhadap performa dan efisiensi.
- Data geospasial harus akurat dan bersih untuk mencegah kesalahan dalam analisis dan query.
- Penulisan query yang efisien agar dapat meningkatkan performa pemrosesan data.
- Penggunaan aplikasi proyeksi dan sistem koordinat yang tepat untuk mendapatkan hasil analisis yang sesuai.

I. Pembagian Kerja

Tabel 9.1 Pembagian Kerja

Nama Anggota	**NIM Anggota**	**Pembagian Kerja**
Louis Caesa Kesuma	13521069	● *Query Definition* ● *Query* DDL ● PPT ● Laporan
Alexander Jason	13521100	● *Data Pre-Processing* ● ERD & *Relational Diagram* ● PPT ● Laporan
Juan Christopher Santoso	13521116	● *Data Pre-Processing* ● Query *DDL* ● Postgis *Implementation* ● Laporan

J. Referensi

PostgreSQL Global Development Group. (n.d.). *PostgreSQL Documentation*. Diakses 1 Desember 2024 dari https://www.postgresql.org/docs/

PostGIS Project. *PostGIS: Spatial and Geographic Objects for PostgreSQL*. Diakses 1 Desember 2024 dari https://postgis.net/

Azizah, Fazat Nur. *Introduction to Spatial Databases*. Diakses 1 Desember 2024 dari https://edunex.itb.ac.id/

Azizah, Fazat Nur. *Physical Representation of Spatial Object*. Diakses 1 Desember 2024 dari https://edunex.itb.ac.id/

Azizah, Fazat Nur. *Spatial Data Modeling*. Diakses 1 Desember 2024 dari https://edunex.itb.ac.id/

Azizah, Fazat Nur. *Spatial Queries and Indexing*. Diakses 1 Desember 2024 dari https://edunex.itb.ac.id/

K. Lampiran

Berikut adalah daftar laman yang digunakan selama pengerjaan tugas besar ini:

- *Repository* GitHub : https://github.com/Gulilil/IF4040_SpatialDatabase
- Google *Drive* : https://drive.google.com/drive/folders/1sHOpGPVtIi7grO8wdq9m345S4rVI66i-?usp=sharing
- *Data Source* :
    - *Country* : https://public.opendatasoft.com/explore/dataset/georef-united-kingdom-region/table/?disjunctive.ctry_code&disjunctive.ctry_name&disjunctive.rgn_code&disjunctive.rgn_name
    - *Region* : https://public.opendatasoft.com/explore/dataset/georef-united-kingdom-region/table/?disjunctive.ctry_code&disjunctive.ctry_name&disjunctive.rgn_code&disjunctive.rgn_name
    - *District* : https://public.opendatasoft.com/explore/dataset/georef-united-kingdom-local-authority-district/table/?disjunctive.ctry_code&disjunctive.ctry_name&disjunctive.rgn_code&disjunctive.rgn_name&disjunctive.ctyua_code&disjunctive.ctyua_name&disjunctive.lad_code&disjunctive.lad_name
    - *Road* : https://public.opendatasoft.com/explore/dataset/europe-road/table/?refine.icc=GB
    - Rail : https://public.opendatasoft.com/explore/dataset/europe-rail-road/table/?refine.icc=GB
    - *Earthquake* : https://earthquakes.bgs.ac.uk/earthquakes/dataSearch.html
- Google *Presentation Slides* : PPT_IF4040_Project3_02
- Link Video: https://youtu.be/PbxSXmLltTI