

Raport

Przetwarzanie i analiza danych przestrzennych

Oracle spatial

Imiona i nazwiska:

Celem ćwiczenia jest zapoznanie się ze sposobem przechowywania, przetwarzania i analizy danych przestrzennych w bazach danych (na przykładzie systemu Oracle spatial)

Swoje odpowiedzi wpisuj w miejsca oznaczone jako:

Wyniki, zrzut ekranu, komentarz

— — —

Do wykonania ćwiczenia (zadania 1 – 7) i wizualizacji danych wykorzystaj Oracle SQL Developer.

Alternatywnie możesz wykonać analizy w środowisku Python/Jupyter Notebook

Do wykonania zadania 8 wykorzystaj środowisko Python/Jupyter Notebook

Raport należy przesłać w formacie pdf.

Należy też dołączyć raport zawierający kod w formacie źródłowym.

Np.

- plik tekstowy .sql z kodem poleceń
- plik .md zawierający kod wersji tekstowej
- notebook programu jupyter – plik .ipynb

Zamieść kod rozwiązania oraz zrzuty ekranu pokazujące wyniki, (dołącz kod rozwiązania w formie tekstowej/źródłowej)

Zwróć uwagę na formatowanie kodu

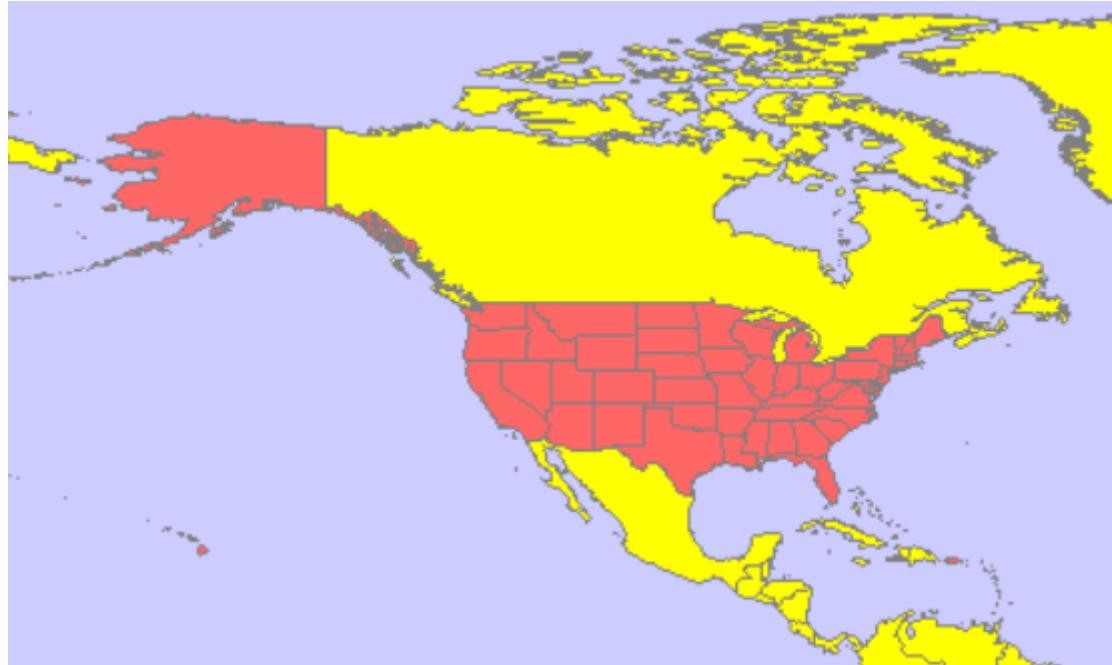
Zadanie 1

Zwizualizuj przykładowe dane

US_STATES

Wyniki, zrzut ekranu, komentarz

```
select * from world_countries;  
select * from us_states;
```



US_INTERSTATES

Wyniki, zrzut ekranu, komentarz

```
select * from us_states;  
select * from us_interstates;
```

Widzimy, że narzędzie nie wyświetla dróg poprawnie. Szybkie przybliżenie i oddalenie wizualizacji przy pomocy myszy odświeża obraz i artefakty znikają.

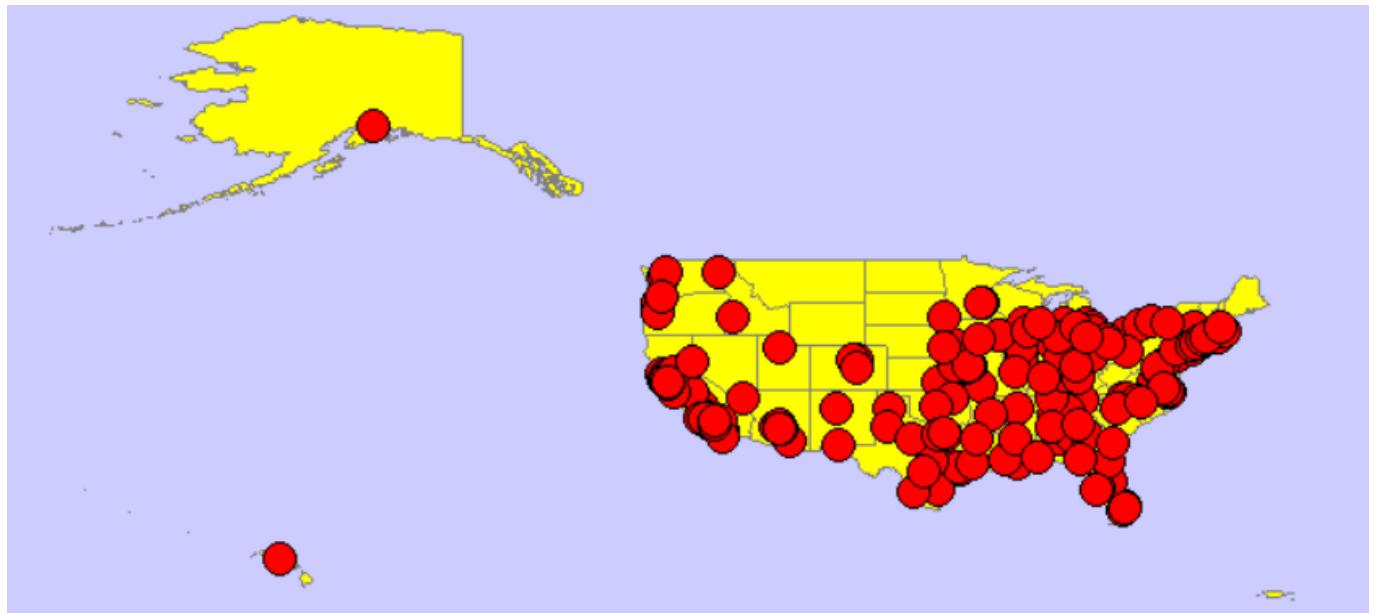


US_CITIES

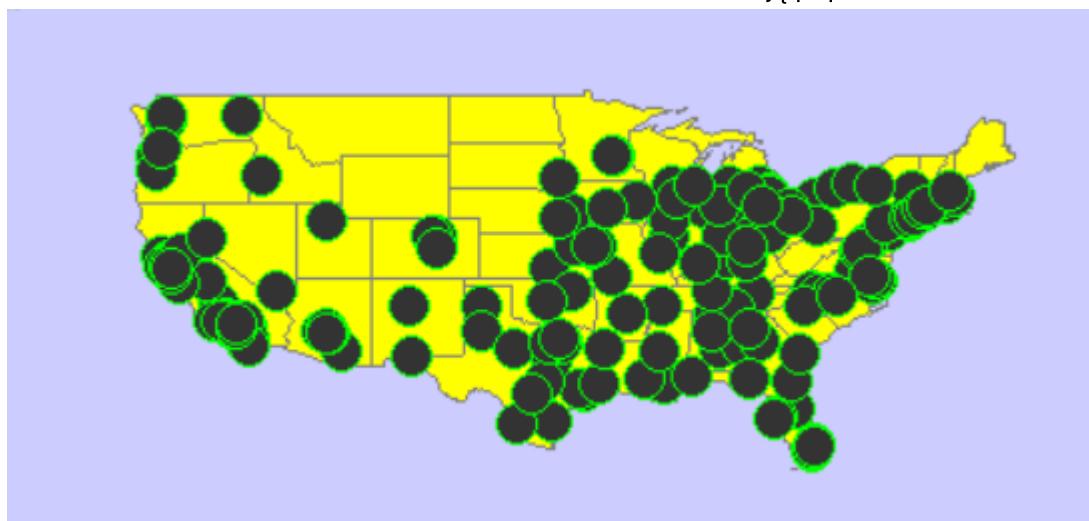
Wyniki, zrzut ekranu, komentarz

```
select * from us_states;
select * from us_cities;
```

Można zauważyć, że markery są bardzo duże. Niestety próby zmiany parametru **Marker width** nie wpływają na rozmiar widocznych markerów.



Marker fill color oraz Marker border color działają poprawnie.



US_RIVERS

Wyniki, zrzut ekranu, komentarz

```
select * from us_states;
select * from us_rivers;
```

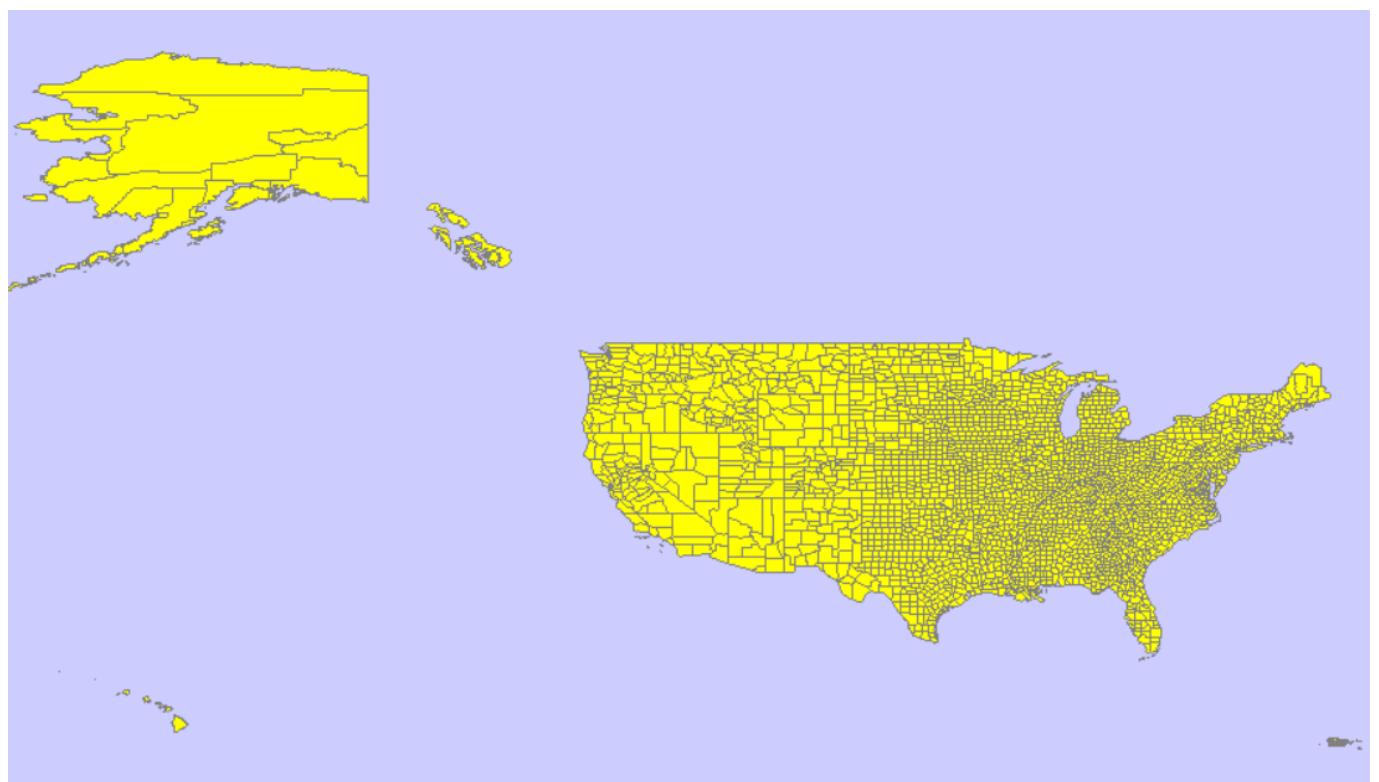


Możemy zaobserwować, że jeziora na północnym wschodzie są oznaczone na czarno mimo, że nie są rzekami. Są to linie w których jeziora te stykają się z granicą Kanady.

US_COUNTIES

Wyniki, zrzut ekranu, komentarz

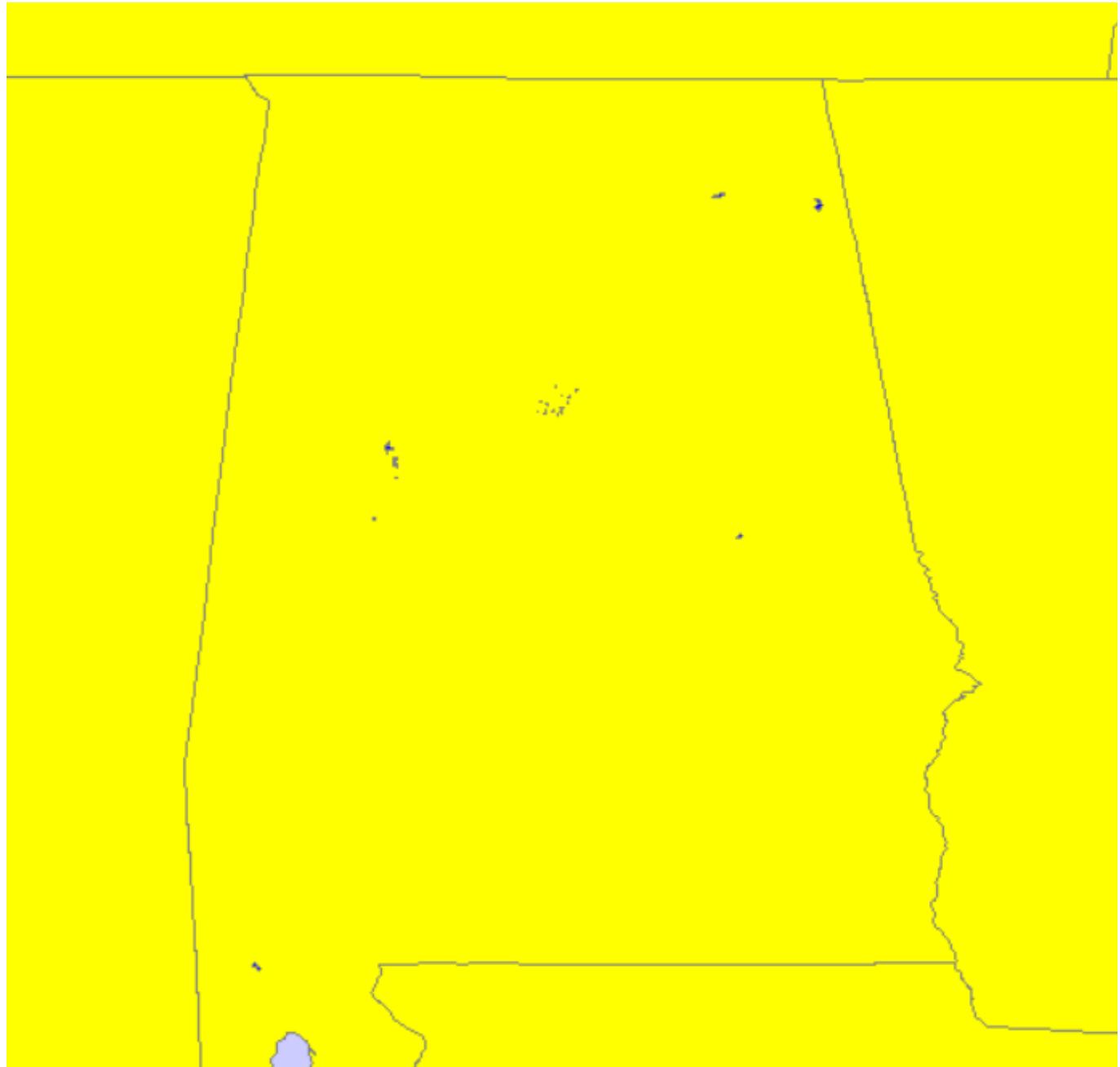
```
select * from us_states;
select * from us_counties;
```



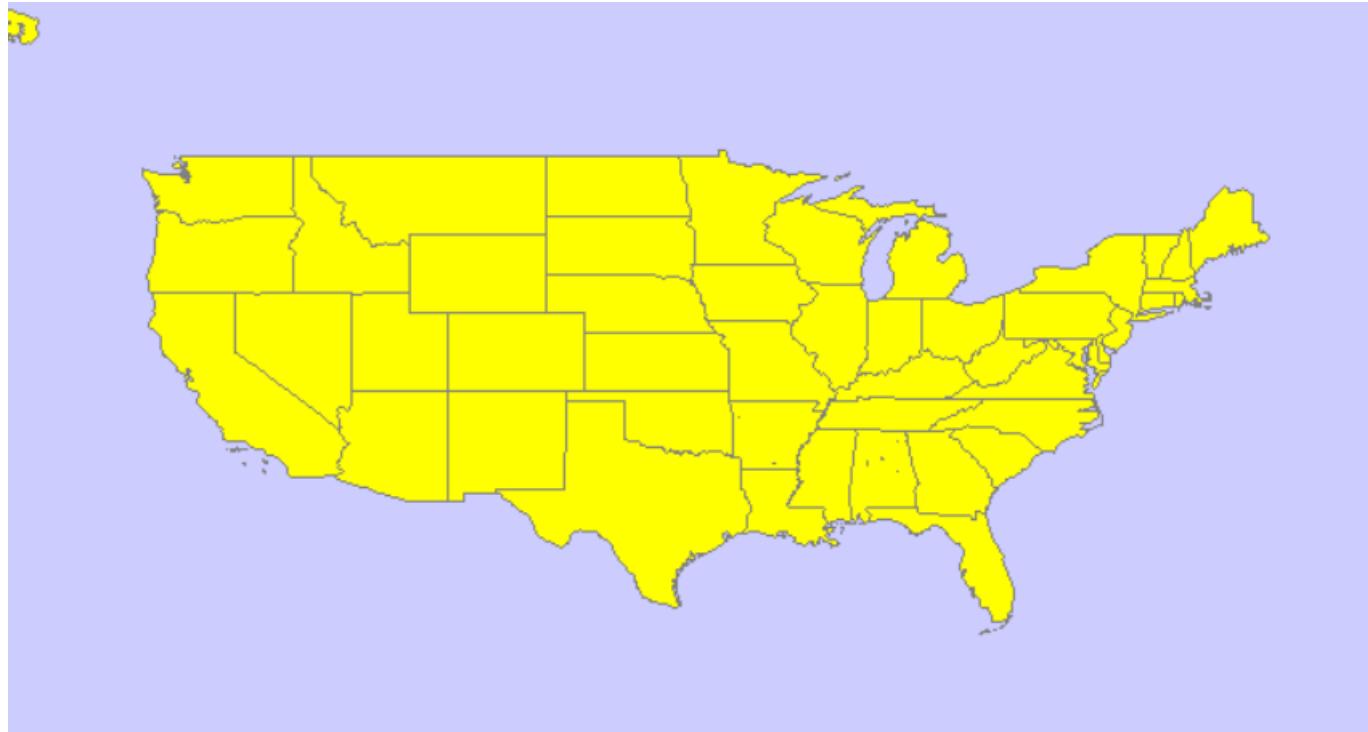
US_PARKS

Wyniki, zrzut ekranu, komentarz

```
select * from us_states;
select * from us_parks
where id < 50;
```



Przez ograniczenie wartości ID do mniejszych niż 50, jesteśmy w stanie zaobserwować głównie parki w stanie Alabama. Domyślnie narzędzie ustawia ich kolor jako żółty, stają się one lepiej widoczne po zmianie koloru.



Parki nie są jednak dobrze widoczne na skali całego kraju.

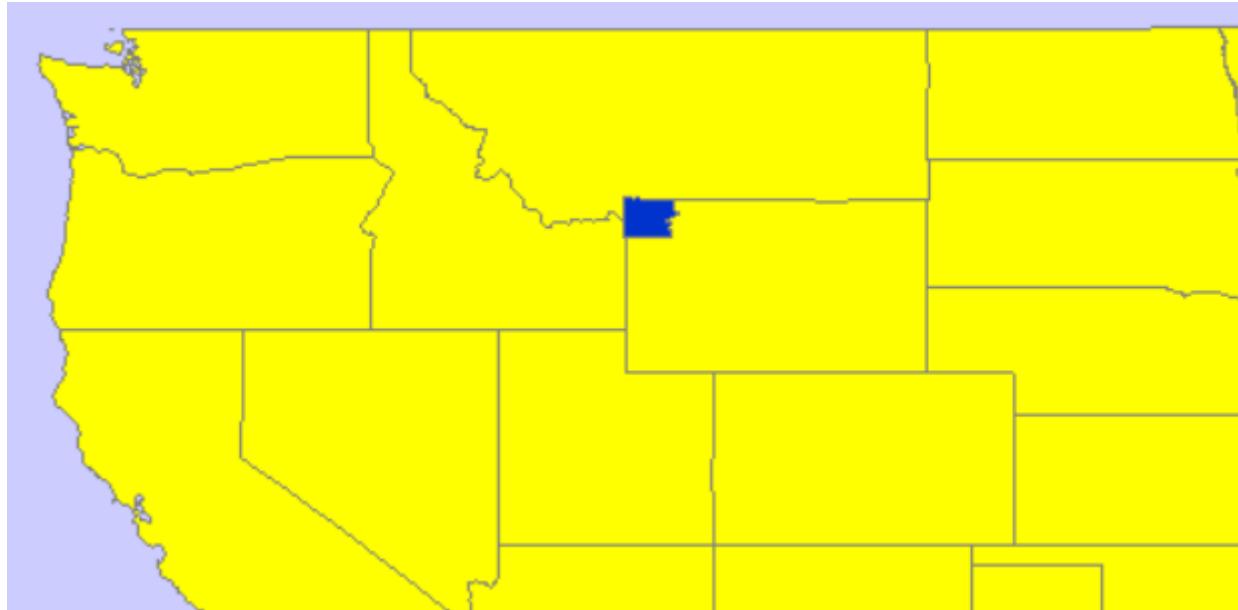
Możemy sprawdzić w tabeli nazwę parku Yellowstone i w ten sposób stworzyć zapytanie, które pozwoli wyświetlić go na mapie

```
select * from us_parks  
order by NAME;
```

ID Yellowstone to 5348

ID	NAME	FCC	GEOM
22	5855 Yellowwood SF	D85	[MDSYS.SDO_GEOMETRY]
23	5348 Yellowstone NP	D83	[MDSYS.SDO_GEOMETRY]
24	5854 Yellow River SF	D85	[MDSYS.SDO_GEOMETRY]
25	4473 Yellow Creek State Park	D85	[MDSYS.SDO_GEOMETRY]

```
select * from us_states;  
select * from us_parks  
where ID = 5348;
```



Zadanie 2

Znajdź wszystkie stany (us_states) których obszary mają część wspólną ze wskazaną geometrią (prostokątem)

Pokaż wynik na mapie.

prostokąt

```
SELECT sdo_geometry (2003, 8307, null,
                     sdo_elem_info_array (1,1003,3),
                     sdo_ordinate_array ( -117.0, 40.0, -90., 44.0)) g
FROM dual
```

Wyniki, zrzut ekranu, komentarz

Wyświetlenie geometrii razem z wynikiem:

The screenshot shows the Oracle SQL Developer interface. At the top, there is a command-line window with the following SQL code:

```
SELECT sdo_geometry (2003, 8307, null,
sdo_elem_info_array (1, 1003, 3),
sdo_ordinate_array(-117.0, 40.0, -90.0, 44.0)) g
FROM dual
```

Below the command-line window is a "Query Result" tab showing the output:

G	(MDSYS.SDO_GEOGRAPHY)
1	{MDSYS.SDO_GEOGRAPHY}

At the bottom of the interface is a spatial map titled "spatial 1". The map displays a single rectangular geometry highlighted in yellow. To the right of the map is a legend configuration panel.

Jest to zwykły prostokąt.

Użyj funkcji SDO_FILTER

```
SELECT state, geom FROM us_states
WHERE sdo_filter (geom,
      sdo_geometry (2003, 8307, null,
      sdo_elem_info_array (1,1003,3),
      sdo_ordinate_array ( -117.0, 40.0, -90., 44.0))
) = 'TRUE';
```

Zwróć uwagę na liczbę zwróconych wierszy (16)

Wyniki, zrzut ekranu, komentarz

```

SELECT state, geom FROM us_states
WHERE sdo_filter (geom,
sdo_geometry (2003, 8307, null,
sdo_elem_info_array (1,1003,3),
sdo_ordinate_array ( -117.0, 40.0, -90., 44.0))
) = 'TRUE';

```

Script Output x Query Result x

All Rows Fetched: 16 in 0.03 seconds

STATE	GEOM
1 Wisconsin	[MDSYS.SDO_GEOMETRY]
2 Illinois	[MDSYS.SDO_GEOMETRY]
3 Michigan	[MDSYS.SDO_GEOMETRY]
4 California	[MDSYS.SDO_GEOMETRY]
5 Oregon	[MDSYS.SDO_GEOMETRY]
6 Nevada	[MDSYS.SDO_GEOMETRY]
7 Idaho	[MDSYS.SDO_GEOMETRY]
8 Utah	[MDSYS.SDO_GEOMETRY]
9 Wyoming	[MDSYS.SDO_GEOMETRY]
10 Colorado	[MDSYS.SDO_GEOMETRY]

Wyswietlenie geometrii z sdo_filter razem z wcześniejsza oraz z wynikiem:

```

SELECT state, geom FROM us_states
WHERE sdo_filter (geom,
sdo_geometry (2003, 8307, null,
sdo_elem_info_array (1, 1003, 3),
sdo_ordinate_array (-117.0, 40.0, -90.0, 44.0))
) = 'TRUE';

```

Script Output x Query Result x

All Rows Fetched: 16 in 0.019 seconds

STATE	GEOM
1 Wisconsin	[MDSYS.SDO_GEOMETRY]
2 Illinois	[MDSYS.SDO_GEOMETRY]
3 Michigan	[MDSYS.SDO_GEOMETRY]
4 California	[MDSYS.SDO_GEOMETRY]
5 Oregon	[MDSYS.SDO_GEOMETRY]
6 Nevada	[MDSYS.SDO_GEOMETRY]
7 Idaho	[MDSYS.SDO_GEOMETRY]
8 Utah	[MDSYS.SDO_GEOMETRY]
9 Wyoming	[MDSYS.SDO_GEOMETRY]
10 Colorado	[MDSYS.SDO_GEOMETRY]
11 Nebraska	[MDSYS.SDO_GEOMETRY]
12 South Dakota	[MDSYS.SDO_GEOMETRY]
13 Kansas	[MDSYS.SDO_GEOMETRY]
14 Iowa	[MDSYS.SDO_GEOMETRY]
15 Minnesota	[MDSYS.SDO_GEOMETRY]
16 Missouri	[MDSYS.SDO_GEOMETRY]

Map View

spatial 1

Active SRID: 8307

Active	Map title	Query	Label column	Label color	Marker type	Marker border color	Marker fill color	Marker width	Curve color	Curve width	Area Line	Area Fill
<input checked="" type="checkbox"/>	SDO_FILTER	SELECT stat...			Circle			8		1		
<input checked="" type="checkbox"/>	SDO geom	SELECT sdo...			Circle			8		1		

Mozemy zauwazyc, ze rzeczywiscie zostało zwrocone 16 wierszy. 2 stany zostały błędnie zakwalifikowane jako pokrywające się z prostokątem. Dokumentacja wskazuje, że funkcja SDO_FILTER, w przeciwieństwie do SDO_ANYINTERACT, służy do szybkiego filtrowania danych, a nie do dokładnego sprawdzania pokrywania się geometrii.

Użyj funkcji SDO_ANYINTERACT

```
SELECT state, geom FROM us_states
WHERE sdo_anyinteract (geom,
    sdo_geometry (2003, 8307, null,
    sdo_elem_info_array (1,1003,3),
    sdo_ordinate_array ( -117.0, 40.0, -90., 44.0))
) = 'TRUE';
```

Porównaj wyniki sdo_filter i sdo_anyinteract

Pokaż wynik na mapie

Wyniki, zrzut ekranu, komentarz

Worksheet Query Builder

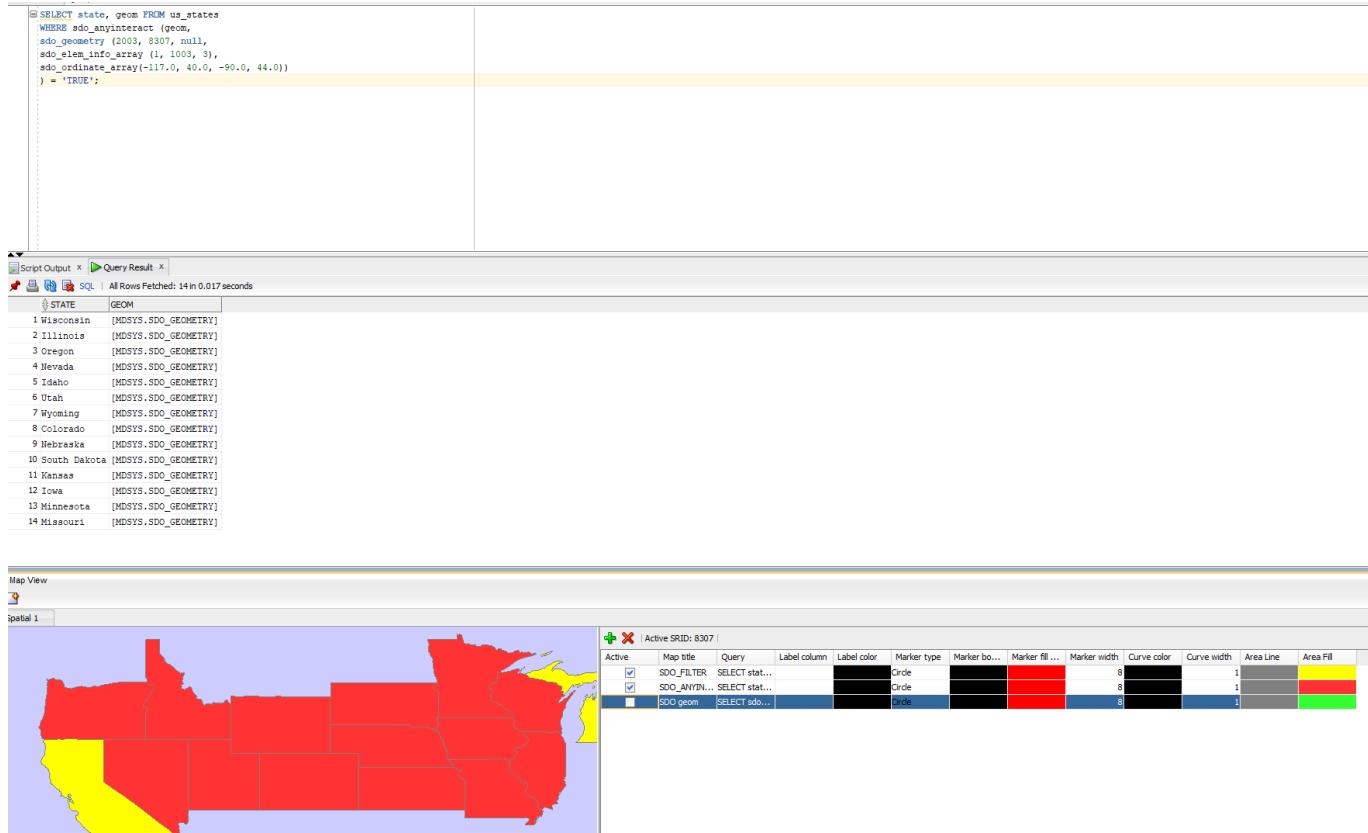
```
SELECT state, geom FROM us_states
WHERE sdo_anyinteract (geom,
    sdo_geometry (2003, 8307, null,
    sdo_elem_info_array (1,1003,3),
    sdo_ordinate_array ( -117.0, 40.0, -90., 44.0))
) = 'TRUE'|
```

Script Output x Query Result x

SQL | All Rows Fetched: 14 in 0.038 seconds

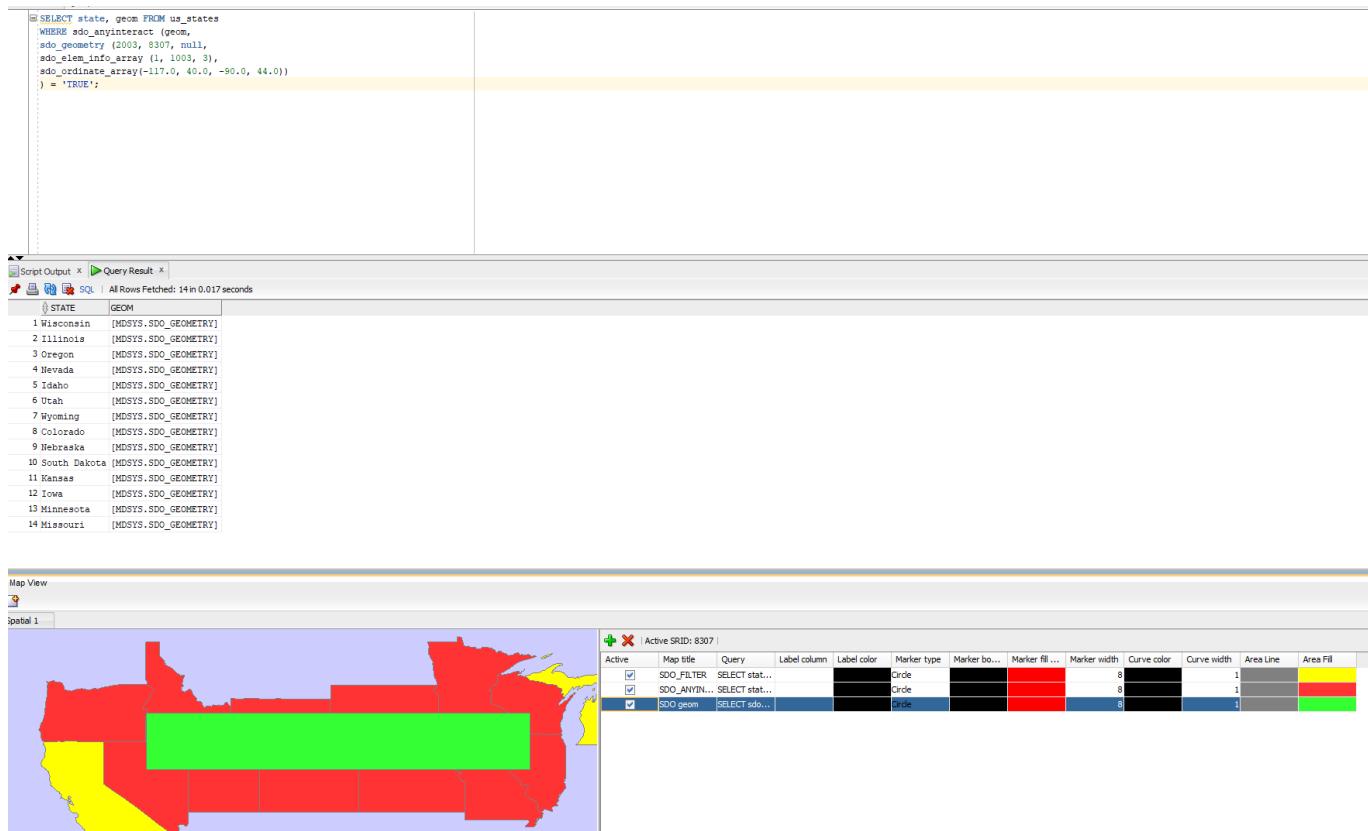
STATE	GEOM
1 Wisconsin	[MDSYS.SDO_GEOMETRY]
2 Illinois	[MDSYS.SDO_GEOMETRY]
3 Oregon	[MDSYS.SDO_GEOMETRY]
4 Nevada	[MDSYS.SDO_GEOMETRY]
5 Idaho	[MDSYS.SDO_GEOMETRY]
6 Utah	[MDSYS.SDO_GEOMETRY]
7 Wyoming	[MDSYS.SDO_GEOMETRY]
8 Colorado	[MDSYS.SDO_GEOMETRY]
9 Nebraska	[MDSYS.SDO_GEOMETRY]
10 South Dakota	[MDSYS.SDO_GEOMETRY]
11 Kansas	[MDSYS.SDO_GEOMETRY]
12 Iowa	[MDSYS.SDO_GEOMETRY]
13 Minnesota	[MDSYS.SDO_GEOMETRY]
14 Missouri	[MDSYS.SDO_GEOMETRY]

Wyswietlenie geometrii otrzymanej z sdo_anyinteract razem z wcześniejszymi oraz z wynikiem:



Zapytanie to zwróciło 14 stanów, co jest poprawna liczbą.

Porównanie wyników obu zapytań, sdo_filter na żółto i sdo_anyinteract na czerwono:



Analiza wyników:

- Zapytanie z użyciem sdo_filter zwróciło 16 wierszy, co jest o 2 więcej niż w przypadku zapytania z użyciem sdo_anyinteract

- W obu wynikach zapytań pokrywa się 14 stanów.
- Patrząc na mapę wyświetlzoną w Oracle SQL Developer, wydaje się, że 2 dodatkowe stany zwrócone przez sdo_filter nie mają punktów wspólnych z geometrią (prostokątem).
- Różnica ta wynika z tego, że funkcja sdo_filter korzysta z **Minimum Bounding Rectangle (MBR)**, czyli wyznacza najmniejszy prostokąt, w który można wpisać daną figurę niebędącą prostokątem, i dopiero na tych prostokątach sprawdza, czy się przecinają. Zaletą tego podejścia jest prostsze i mniej kosztowne obliczanie (szczególnie dla skomplikowanych kształtów), dlatego jest stosowane do wstępnego filtrowania danych. Wadą jest zwracanie fałszywie pozytywnych wyników, tak jak w naszym przypadku z dwoma dodatkowymi stanami.

Zadanie 3

Znajdź wszystkie parki (us_parks) których obszary znajdują się wewnątrz stanu Wyoming

Użyj funkcji SDO_INSIDE

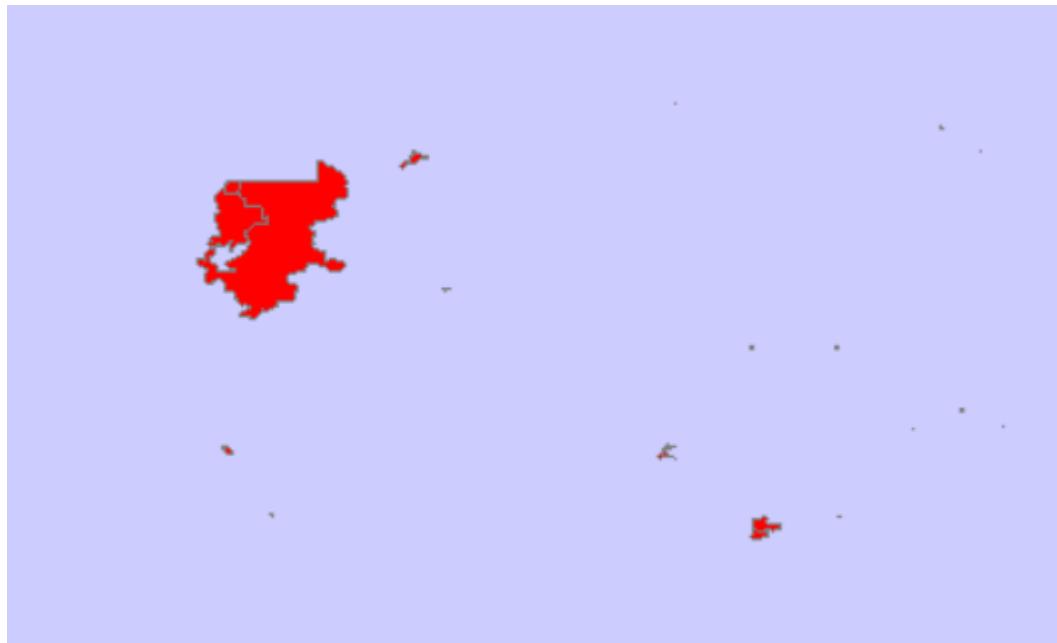
```
SELECT p.name, p.geom
FROM us_parks p,
     us_states s
WHERE s.state = 'Wyoming'
      AND SDO_INSIDE (p.geom, s.geom ) = 'TRUE';
```

W przypadku wykorzystywania narzędzia SQL Developer, w celu wizualizacji na mapie użyj podzapytania

```
SELECT pp.name, pp.geom  FROM us_parks pp
WHERE id IN
(
    SELECT p.id
    FROM us_parks p, us_states s
    WHERE s.state = 'Wyoming'
        and SDO_INSIDE (p.geom, s.geom ) = 'TRUE'
)
```

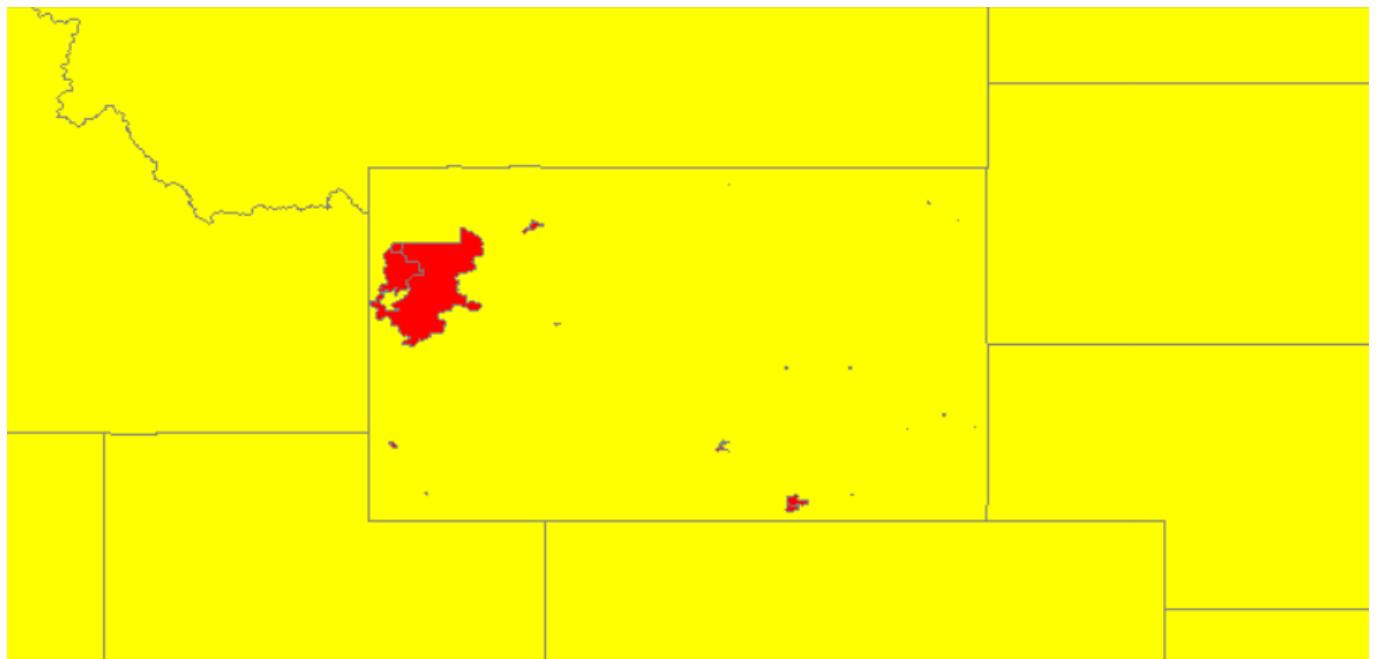
Wyniki, zrzut ekranu, komentarz

Same parki z podzapytania nie dają nam dużo informacji.



Wyświetlmy je na mapie USA.

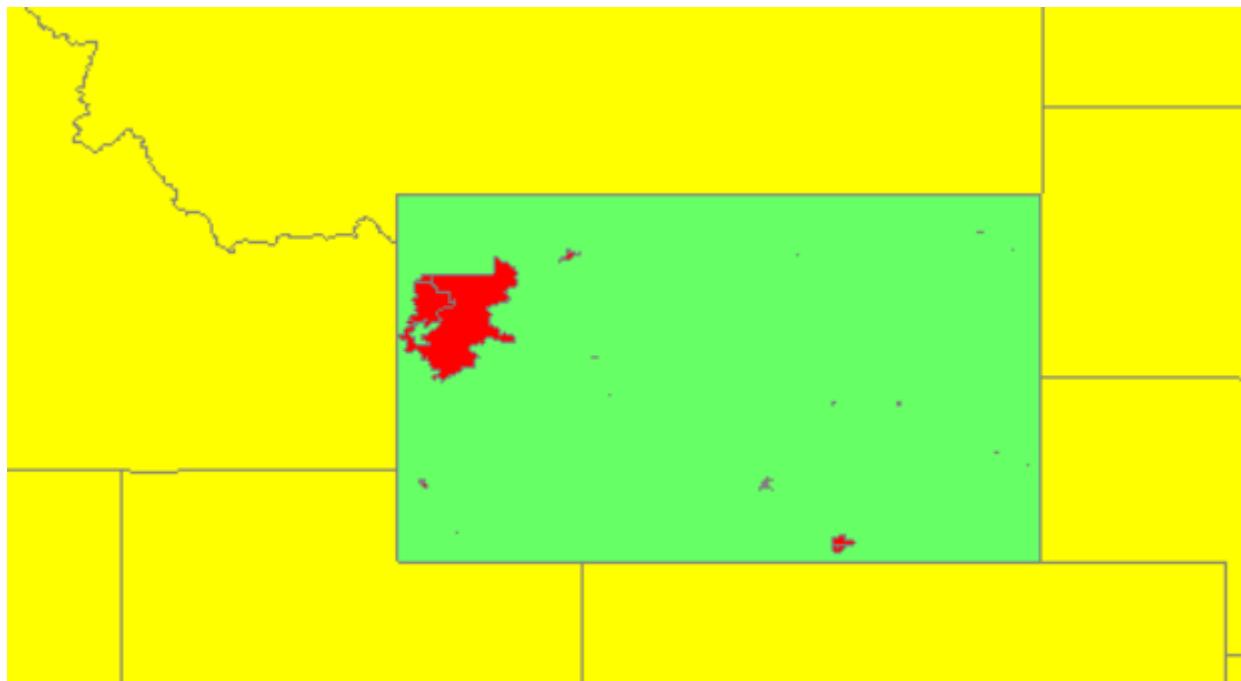
```
select * from us_states;
```



Dodajmy stan Wyoming i zaznaczmy go innym kolorem w celu rozróżnienia.

```
SELECT state, geom FROM us_states  
WHERE state = 'Wyoming'
```

Wyniki, zrzut ekranu, komentarz



Porównaj wynik z:

```
SELECT p.name, p.geom
FROM us_parks p, us_states s
WHERE s.state = 'Wyoming'
AND SDO_ANYINTERACT (p.geom, s.geom) = 'TRUE';
```

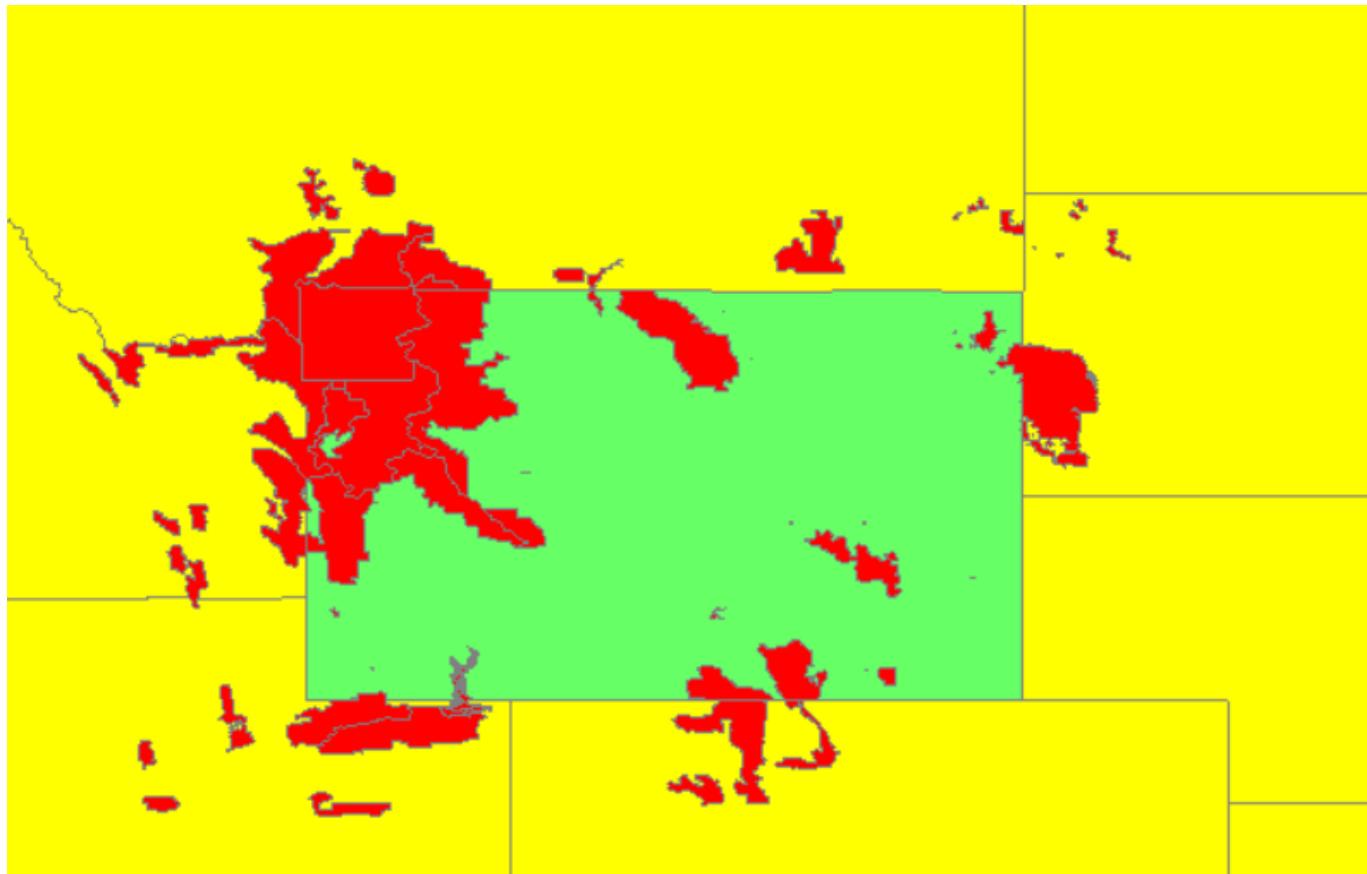
W celu wizualizacji użyj podzapytania

Wyniki, zrzut ekranu, komentarz

Podzapytanie umożliwiające wizualizację:

```
SELECT pp.name, pp.geom FROM us_parks pp
WHERE id IN
(
    SELECT p.id
    FROM us_parks p, us_states s
    WHERE s.state = 'Wyoming'
    AND SDO_ANYINTERACT (p.geom, s.geom) = 'TRUE'
)
```

Widzimy, dużo więcej parków. Na przykład widoczny park Yellowstone wcześniej nie był widoczny na mapie. Można wyciągnąć wniosek, że funkcja **SDO_INSIDE** pozwala na wyodrębnienie tylko tych elementów geometrycznych, które w całości znajdują się wewnątrz wybranego obszaru i nie dotykają jego granic. Za to **SDO_ANYINTERACT** wyodrębnia też obszary częściowo nachodzące na wybrany oraz te przyległe do niego.



Zadanie 4

Znajdź wszystkie jednostki administracyjne (us_counties) wewnętrz stanu New Hampshire

```
SELECT c.county, c.state_abrv, c.geom
FROM us_counties c, us_states s
WHERE s.state = 'New Hampshire'
AND SDO_RELATE ( c.geom,s.geom, 'mask=INSIDE+COVEREDBY' ) = 'TRUE';

SELECT c.county, c.state_abrv, c.geom
FROM us_counties c, us_states s
WHERE s.state = 'New Hampshire'
AND SDO_RELATE ( c.geom,s.geom, 'mask=INSIDE' ) = 'TRUE';

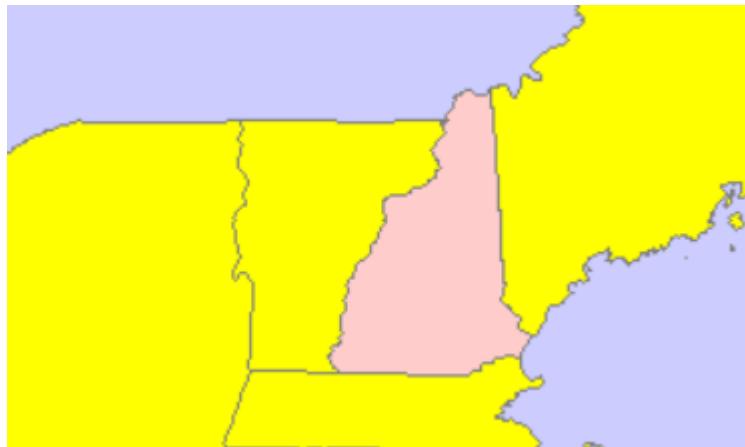
SELECT c.county, c.state_abrv, c.geom
FROM us_counties c, us_states s
WHERE s.state = 'New Hampshire'
AND SDO_RELATE ( c.geom,s.geom, 'mask=COVEREDBY' ) = 'TRUE';
```

W przypadku wykorzystywania narzędzia SQL Developer, w celu wizualizacji danych na mapie należy użyć podzapytania (podobnie jak w poprzednim zadaniu)

Wyniki, zrzut ekranu, komentarz

Mapa z fragmentem na którym widać New Hampshire.

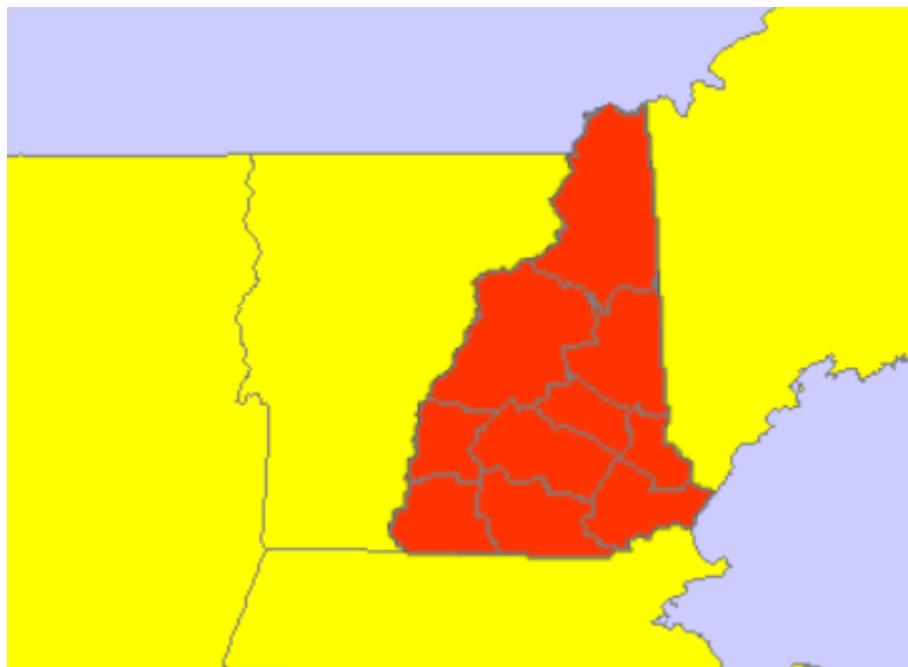
```
Select * from us_states  
WHERE state = 'New Hampshire';
```



Zobaczmy jak wygląda wynik pierwszego zapytania - z maską INSIDE+COVEREDBY

```
SELECT cc.county, cc.state_abrv, cc.geom FROM us_counties cc  
WHERE id IN  
(  
    SELECT c.id  
    FROM us_counties c, us_states s  
    WHERE s.state = 'New Hampshire'  
    AND SDO_RELATE ( c.geom,s.geom, 'mask=INSIDE+COVEREDBY' ) = 'TRUE'  
);
```

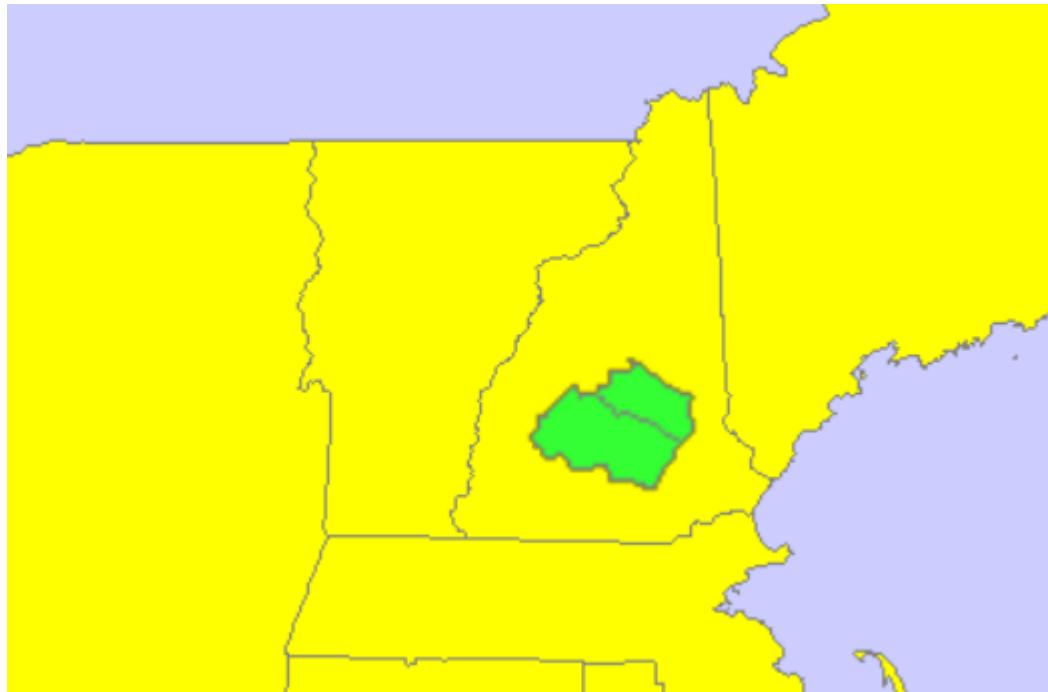
Widzimy, że cały stan jest pokryty fragmentami - hrabstwami.



Sprawdźmy drugie zapytanie - maskę INSIDE.

```
SELECT cc.county, cc.state_abrv, cc.geom FROM us_counties cc
WHERE id IN
(
    SELECT c.id
    FROM us_counties c, us_states s
    WHERE s.state = 'New Hampshire'
    AND SDO_RELATE ( c.geom,s.geom, 'mask=INSIDE' ) = 'TRUE'
);
```

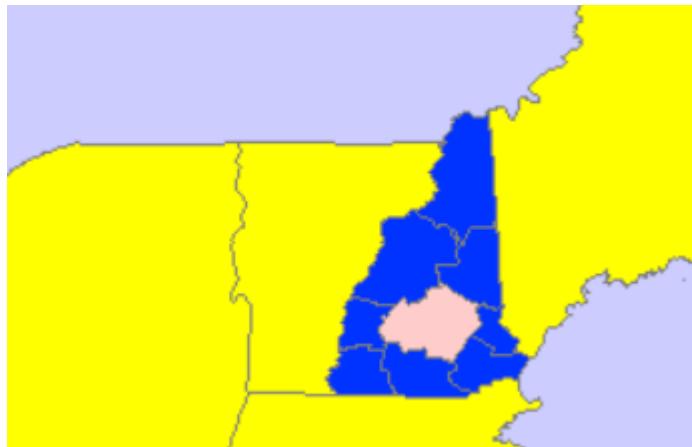
INSIDE powoduje wybranie tylko tych hrabstw które w całości znajdują się w stanie.



Trzecie zapytanie - maska COVEREDBY

```
SELECT cc.county, cc.state_abrv, cc.geom FROM us_counties cc
WHERE id IN
(
    SELECT c.id
    FROM us_counties c, us_states s
    WHERE s.state = 'New Hampshire'
    AND SDO_RELATE ( c.geom,s.geom, 'mask=COVEREDBY' ) = 'TRUE'
);
```

Widzimy, że tylko te hrabstwa które stykają się z granicą stanu zostają wybrane



Z obserwacji można wyciągnąć wnioski o masce **COVEREDBY** - nie uwzględnia ona elementów całkowicie zawartych wewnątrz stanu New Hampshire. Oznacza to, że żeby coś zostało przez nią uwzględnione musi mieć część wspólną z obszarem na zewnątrz stanu. W tym przypadku tą częścią wspólną jest granica.

Zadanie 5

Znajdź wszystkie miasta w odległości 50 mili od drogi (us_interstates) I4

Pokaż wyniki na mapie

```
SELECT * FROM us_interstates
WHERE interstate = 'I4'

SELECT * FROM us_states
WHERE state_abrv = 'FL'

SELECT c.city, c.state_abrv, c.location
FROM us_cities c
WHERE ROWID IN
(
  SELECT c.rowid
  FROM us_interstates i, us_cities c
  WHERE i.interstate = 'I4'
  AND sdo_within_distance (c.location, i.geom,'distance=50 unit=mile'
)
```

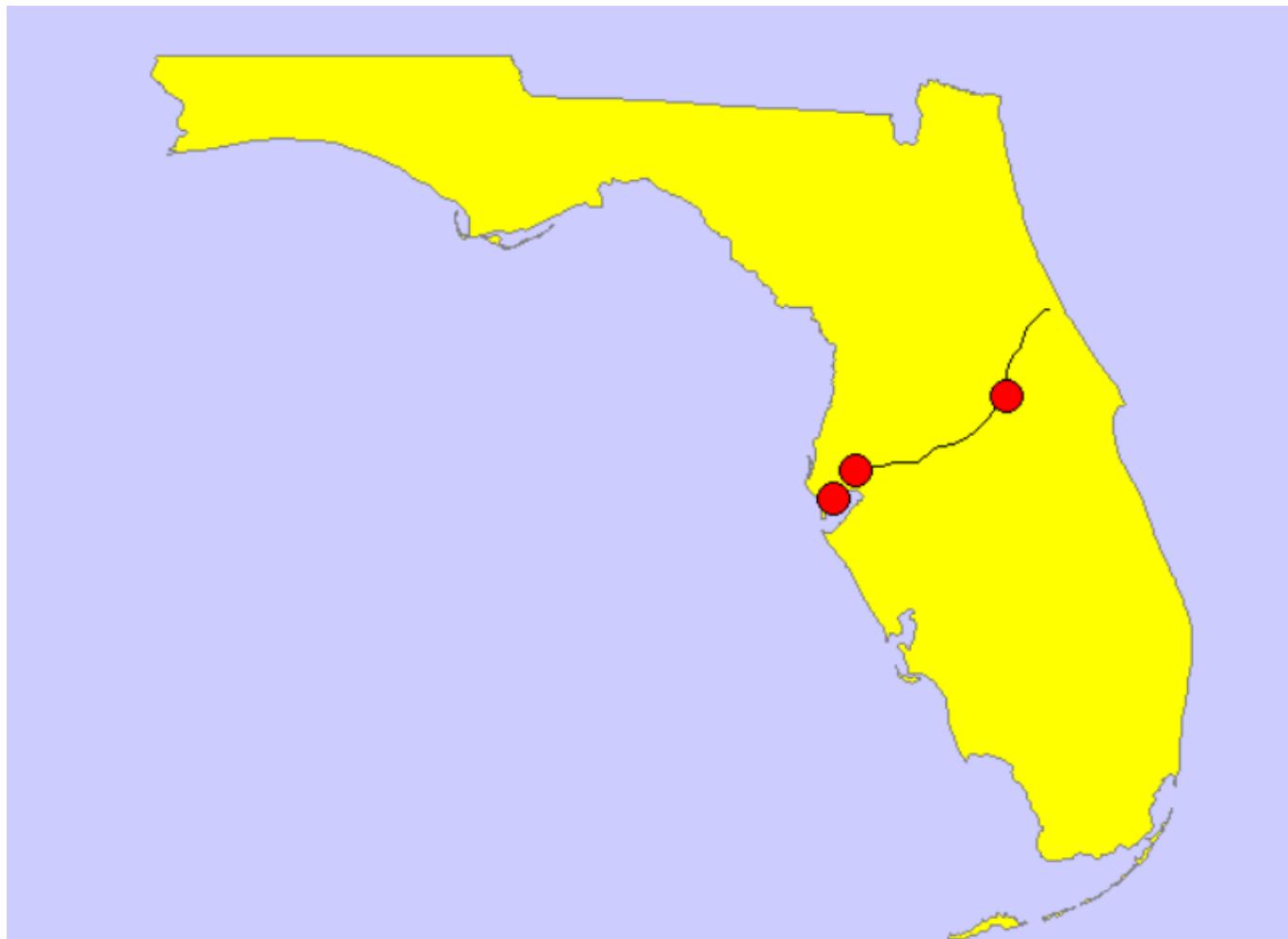
Wyniki, zrzut ekranu, komentarz

Musieliśmy zmodyfikować ostatnie zapytanie:

```
SELECT c.city, c.state_abrv, c.location
FROM us_cities c
WHERE ROWID IN(
  SELECT c.rowid
  FROM us_interstates i, us_cities c
  WHERE i.interstate = 'I4'
```

```
AND sdo_within_distance(c.location, i.geom, 'distance=50 unit=mile') =  
'TRUE')
```

Na wizualizacji możemy zobaczyć wyraźnie drogę i trzy miasta w odpowiedniej odległości

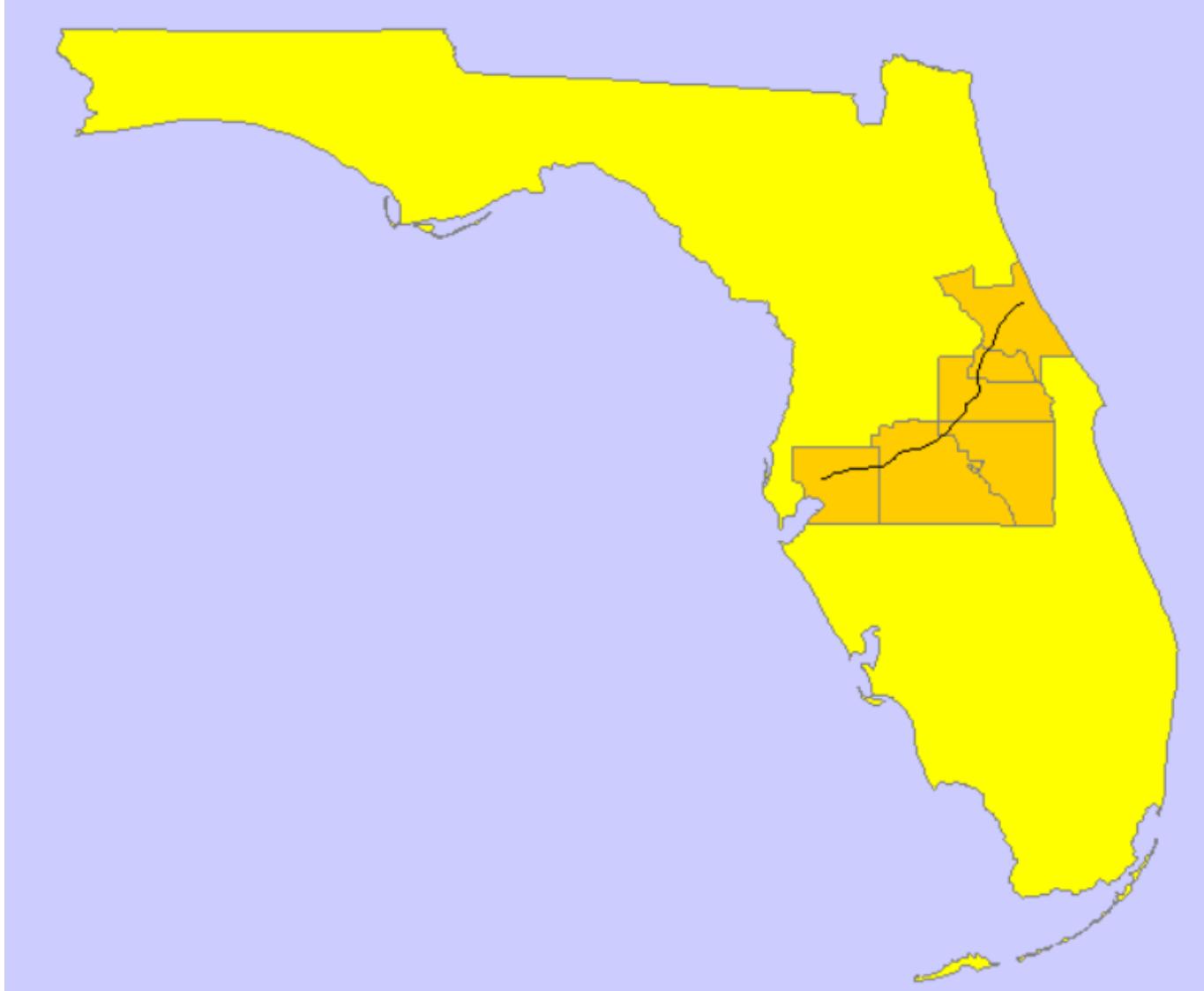


Dodatkowo:

- Znajdz wszystkie jednostki administracyjne przez które przechodzi droga I4

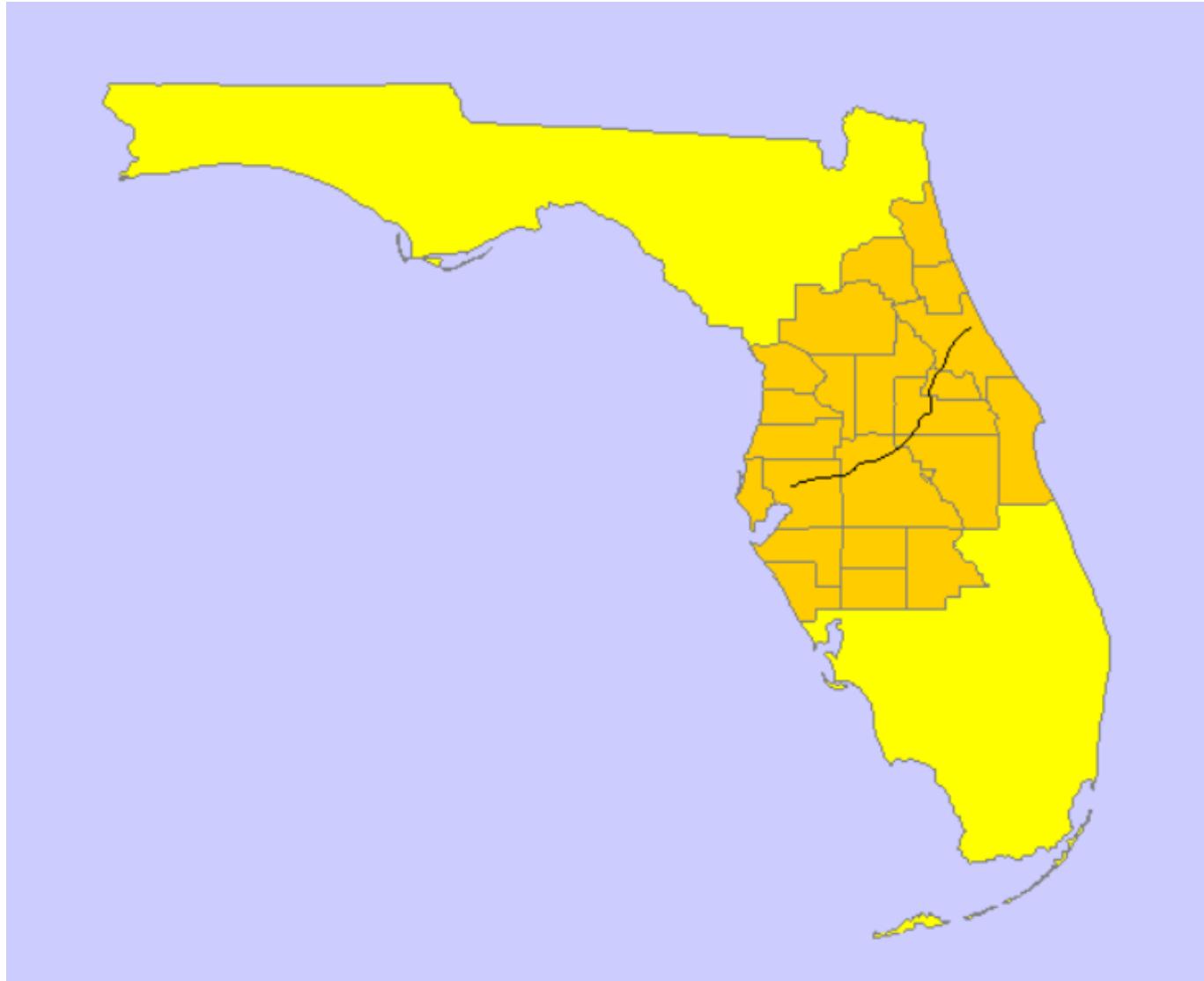
W celu określenia wszystkich jednostek administracyjnych przez które przechodzi droga wystarczy zmniejszyć dystans do wartości 0.

```
SELECT c.county, c.geom  
FROM us_counties c  
WHERE c.state_abrv = 'FL'  
and c.id IN(  
SELECT c.id  
FROM us_interstates i, us_counties c  
WHERE i.interstate = 'I4'  
AND sdo_within_distance(c.geom, i.geom, 'distance=0 unit=mile') = 'TRUE'  
)
```



- b) Znajdz wszystkie jednostki administracyjne w pewnej odleglosci od I4

```
SELECT c.county, c.geom
FROM us_counties c
WHERE c.state_abrv = 'FL'
and c.id IN(
SELECT c.id
FROM us_interstates i, us_counties c
WHERE i.interstate = 'I4'
AND sdo_within_distance(c.geom, i.geom, 'distance=50 unit=mile') = 'TRUE'
)
```

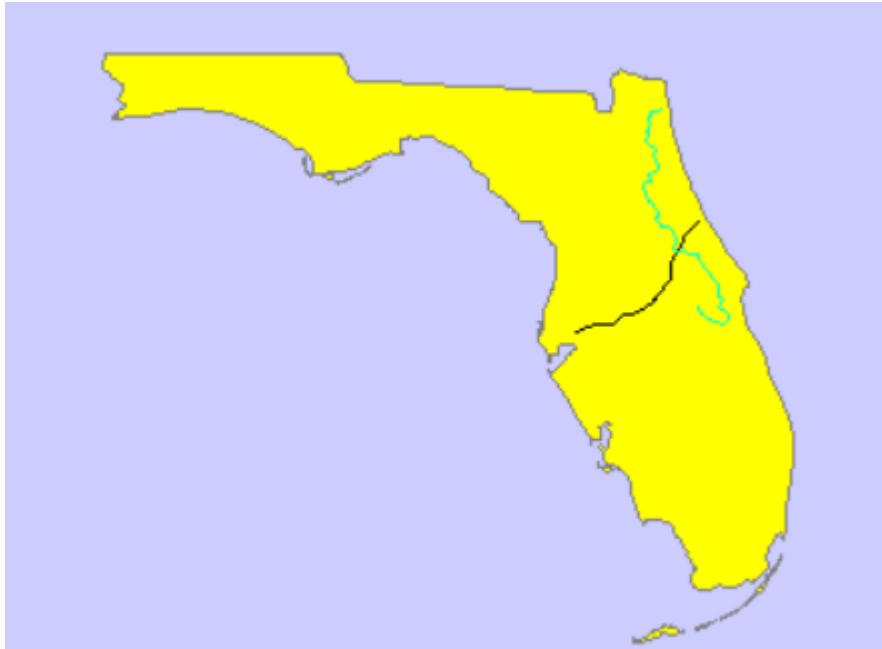


c) Znajdz rzeki które przecina droga I4

Możemy zastosować podejście z podpunktu a)

```
SELECT r.name, r.geom
FROM us_rivers r
WHERE r.id IN(
  SELECT r.id
  FROM us_interstates i, us_rivers r
  WHERE i.interstate = 'I4'
  AND sdo_within_distance(r.geom, i.geom, 'distance=0 unit=mile') = 'TRUE'
)
```

St. Jones jest jedyną rzeką która przecina się z drogą I4



- d) Znajdz wszystkie drogi które przecinają rzekę Mississippi

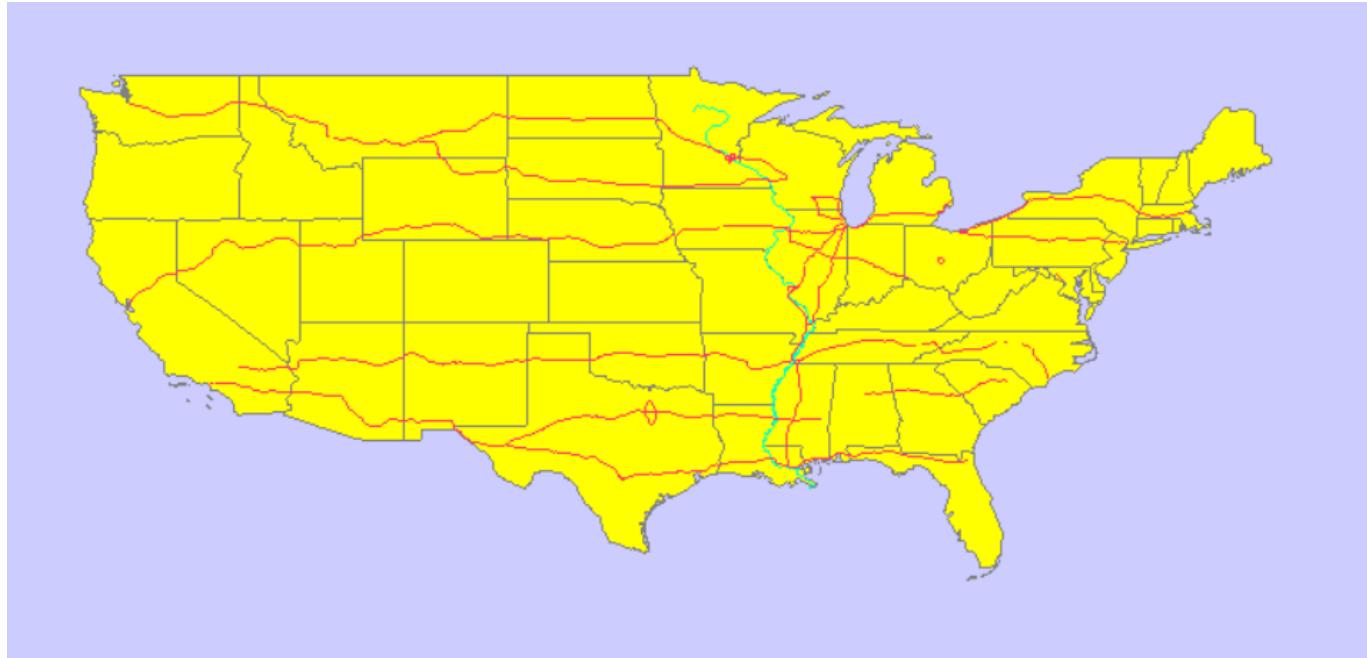
W wyborze stanów do wizualizacji pomijamy Alaskę.

```
SELECT * FROM us_states
where state != 'Alaska';

SELECT * FROM us_rivers
WHERE name = 'Mississippi';

SELECT i.interstate, i.geom
FROM us_interstates i
WHERE i.id IN(
SELECT i.id
FROM us_interstates i, us_rivers r
WHERE r.name = 'Mississippi'
AND sdo_within_distance(r.geom, i.geom, 'distance=0 unit=mile') = 'TRUE'
);
```

Widzimy, że drogi które przecinają Mississippi rozciągają się na całą szerokość kraju.



Niektóre drogi wyglądają jakby nie przecinały się z rzeką, ale kiedy najedziemy myszą na taką drogę możemy zobaczyć, że jest to poprostu nieciągłość między jej odcinkami.



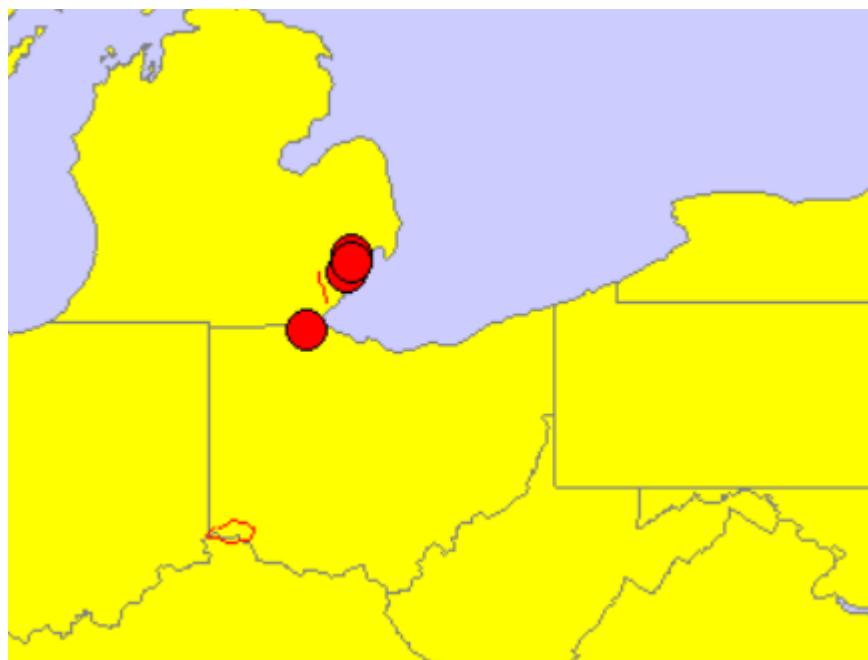
- e) Znajdz wszystkie miasta w odległości od 15 do 30 mil od drogi 'I275'

Funkcji `sdo_within_distance` nie można użyć z opcją '`FALSE`' więc poprostu zaprzeczamy jej użycie z '`TRUE`'

```
SELECT c.city, c.state_abrv, c.location
FROM us_cities c
WHERE ROWID IN(
  SELECT c.rowid
  FROM us_interstates i, us_cities c
    WHERE sdo_within_distance(i.location, c.location, 30) = 1
      AND sdo_within_distance(i.location, c.location, 15) = 0)
```

```
WHERE i.interstate = 'I275'  
  AND sdo_within_distance(c.location, i.geom, 'distance=30 unit=mile') =  
  'TRUE'  
  AND not sdo_within_distance(c.location, i.geom, 'distance=15 unit=mile')  
= 'TRUE'  
)
```

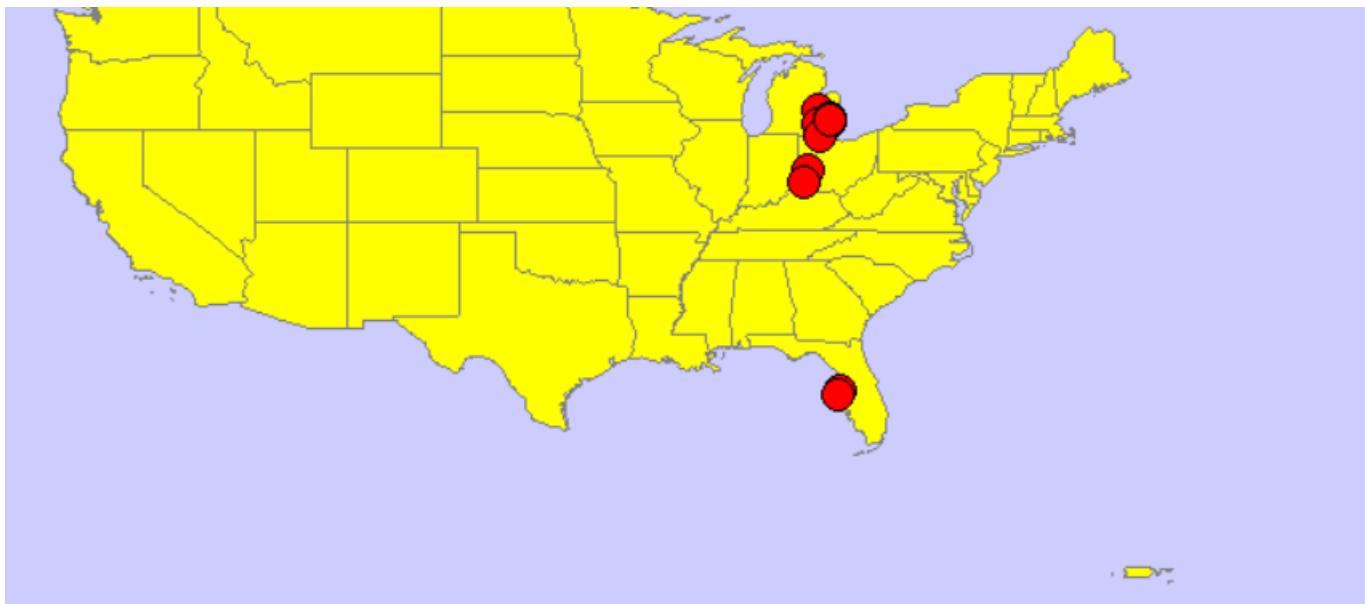
Widzimy, że droga I275 ma więcej niż jeden fragment, ale tylko przy północnym możemy znaleźć miasta o zadanych odległościach



f) Sprawdźmy czy znajdziemy więcej miast dla podpunktu e) ustawiając inną odległość od drogi.

```
SELECT c.city, c.state_abrv, c.location  
FROM us_cities c  
WHERE ROWID IN(  
SELECT c.rowid  
FROM us_interstates i, us_cities c  
WHERE i.interstate = 'I275'  
AND sdo_within_distance(c.location, i.geom, 'distance=60 unit=mile') =  
'TRUE'  
)
```

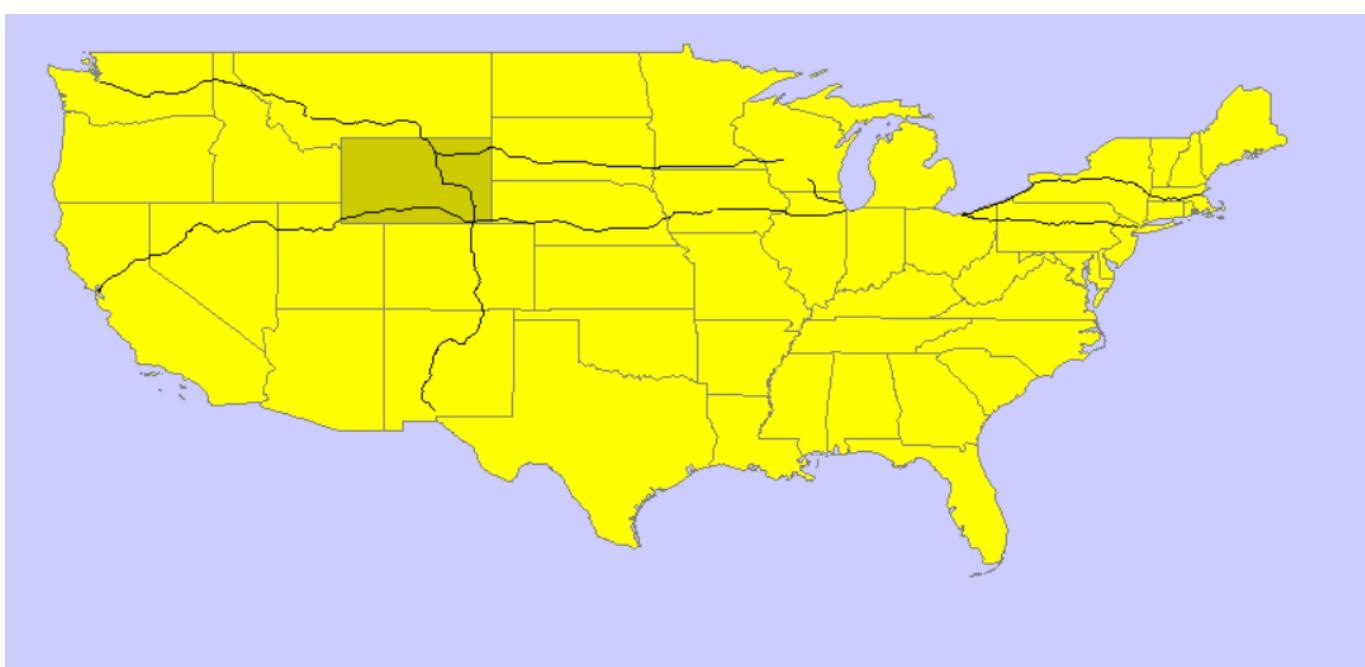
Okazuje się, że na Florydzie też jest fragment drogi I275



g) Wszystkie drogi które przechodzą przez stan Wyoming

```
select * from us_states
where state_abrv = 'WY';

SELECT i.interstate, i.geom
FROM us_interstates i
WHERE i.id IN(
SELECT i.id
FROM us_interstates i, us_states s
WHERE s.state_abrv = 'WY'
AND sdo_within_distance(s.geom, i.geom, 'distance=0 unit=mile') = 'TRUE'
);
```



Zadanie 6

Znajdz 5 miast najbliższych drogi I4

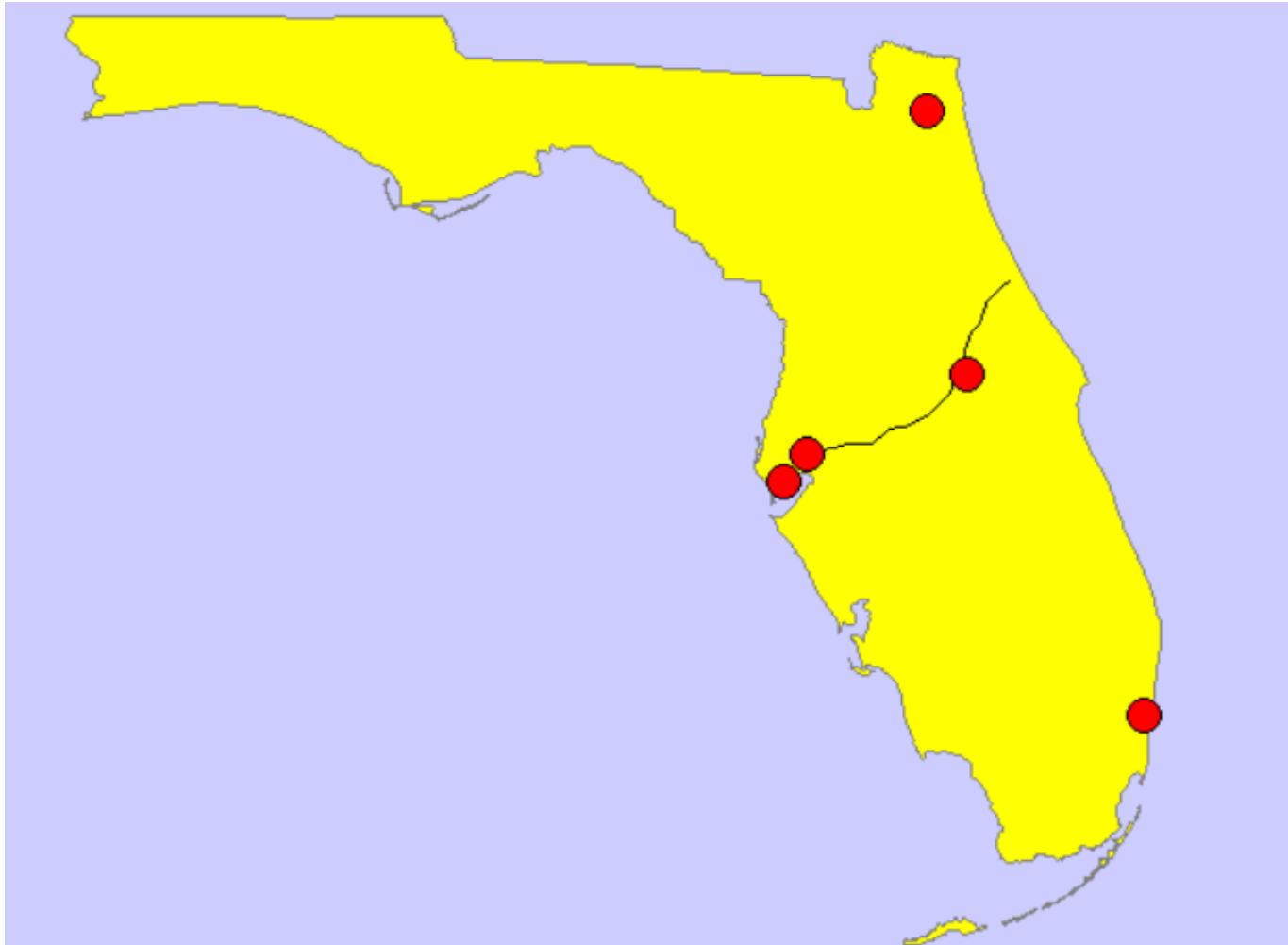
```
SELECT c.city, c.state_abrv, c.location
FROM us_interstates i, us_cities c
WHERE i.interstate = 'I4'
AND sdo_nn(c.location, i.geom, 'sdo_num_res=5') = 'TRUE';
```

Wyniki, zrzut ekranu, komentarz

```
select * from us_states
where state_abrv = 'FL';

select * from us_interstates
where interstate = 'I4';

SELECT c.city, c.state_abrv, c.location
FROM us_cities c
WHERE c.id in (
    select c.id
    FROM us_interstates i, us_cities c
    WHERE i.interstate = 'I4'
    AND sdo_nn(c.location, i.geom, 'sdo_num_res=5') = 'TRUE'
);
```



Dodatkowo:

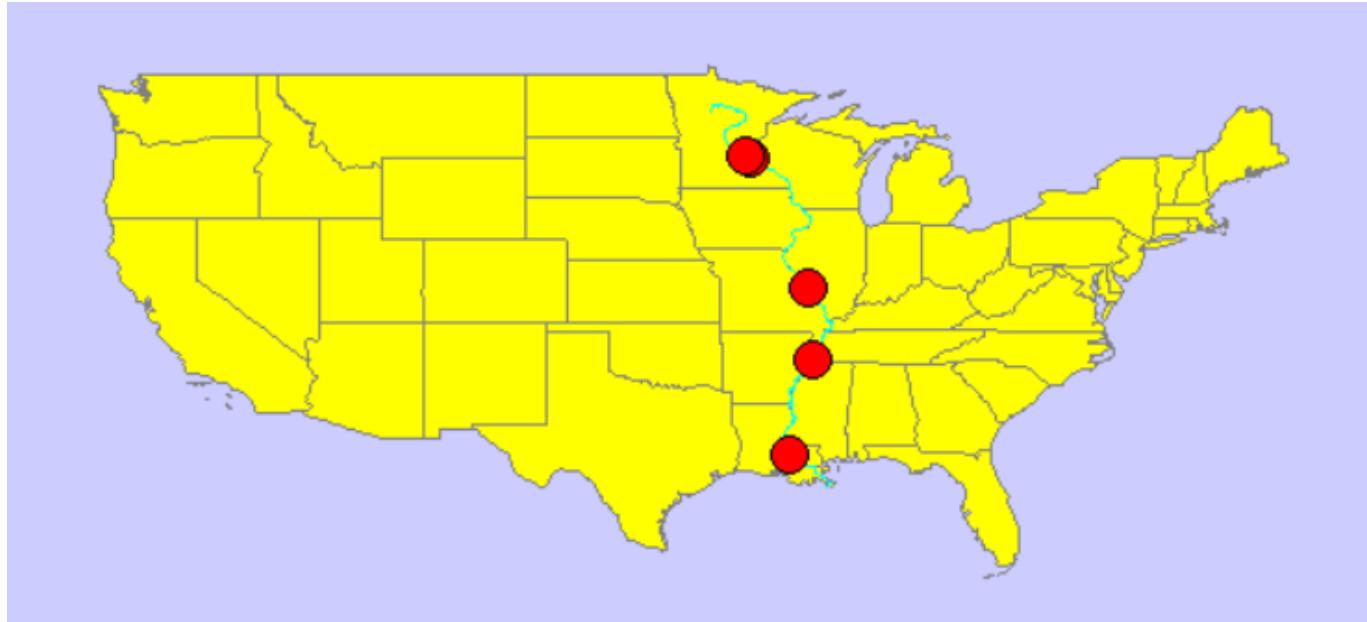
a) Znajdz kilka miast najbliszzych rzece Mississippi

```
select * from us_states
where state_abrv != 'AK';

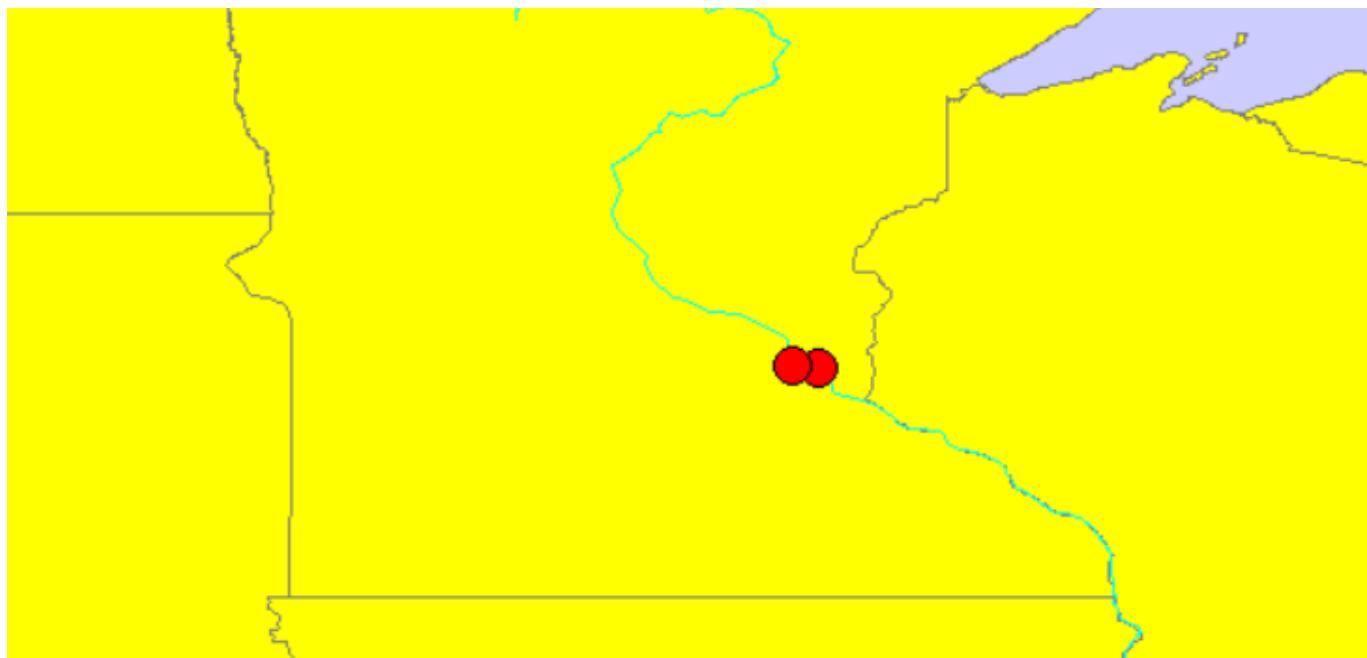
select * from us_rivers
where name = 'Mississippi';

SELECT c.city, c.state_abrv, c.location
FROM us_cities c
WHERE c.id in (
    select c.id
    FROM us_rivers r, us_cities c
    WHERE r.name = 'Mississippi'
    AND sdo_nn(c.location, r.geom, 'sdo_num_res=5') = 'TRUE'
);
```

Na pierwszy rzut oka wygląda na to, że są tylko 4 miasta. Czy to błąd?



Nie! 2 z miast leżą poprostu blisko siebie



Znalezione miasta to St Paul, Memphis, St Louis, Minneapolis oraz Baton Rouge

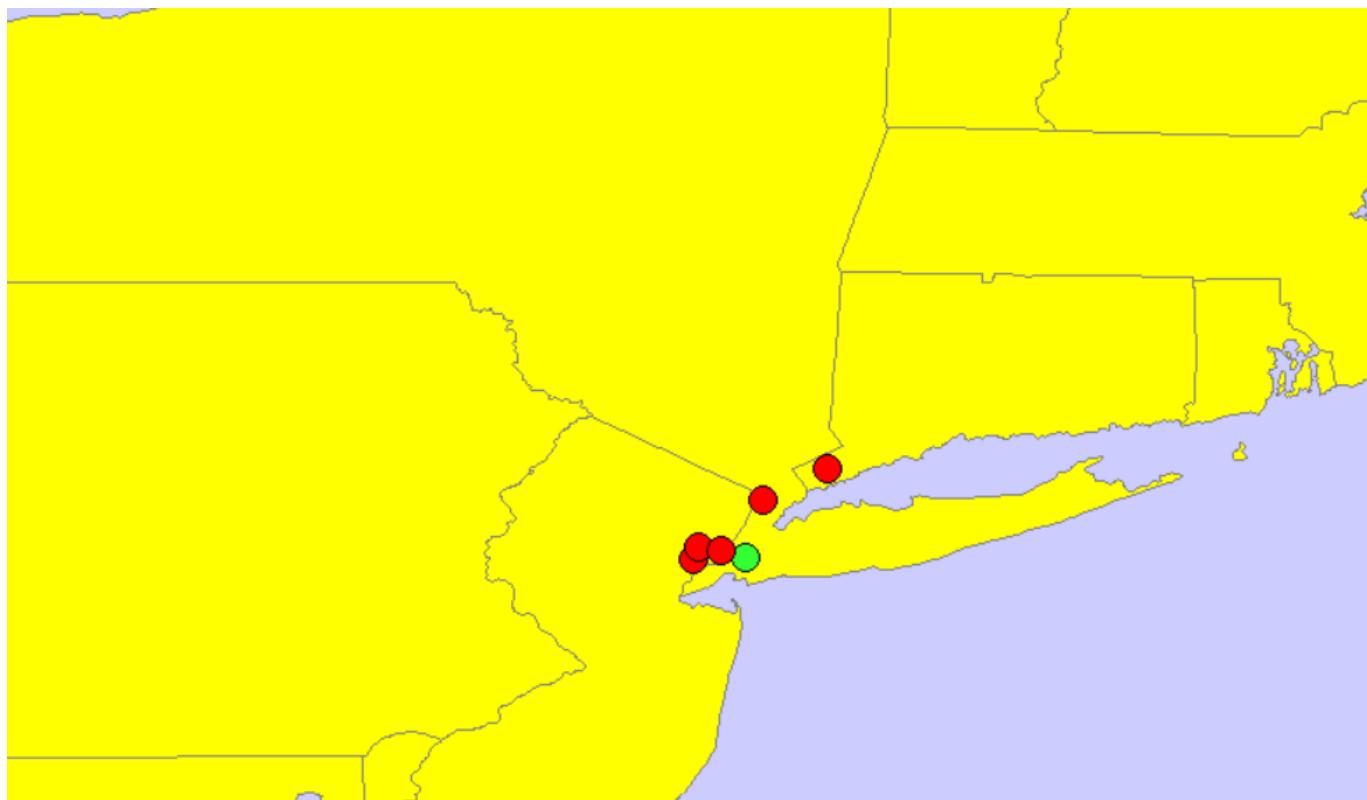
b) Znajdz 3 miasta najbliżej Nowego Jorku

Należy wyświetlić wszystkie stany - część z najbliższych miast znajduje się już poza

```
select * from us_states  
where state_abrv != 'AK';  
  
select * from us_cities  
where city = 'New York';  
  
SELECT c.city, c.state_abrv, c.location  
FROM us_cities c  
WHERE c.id in (
```

```
select c2.id
FROM us_cities c,
(select * from us_cities
where city != 'New York') c2
WHERE c.city = 'New York'
AND sdo_nn(c.location, c2.location, 'sdo_num_res=5') = 'TRUE'
);
```

Nowy Jork zaznaczyliśmy innym kolorem dla lepszego rozróżnienia



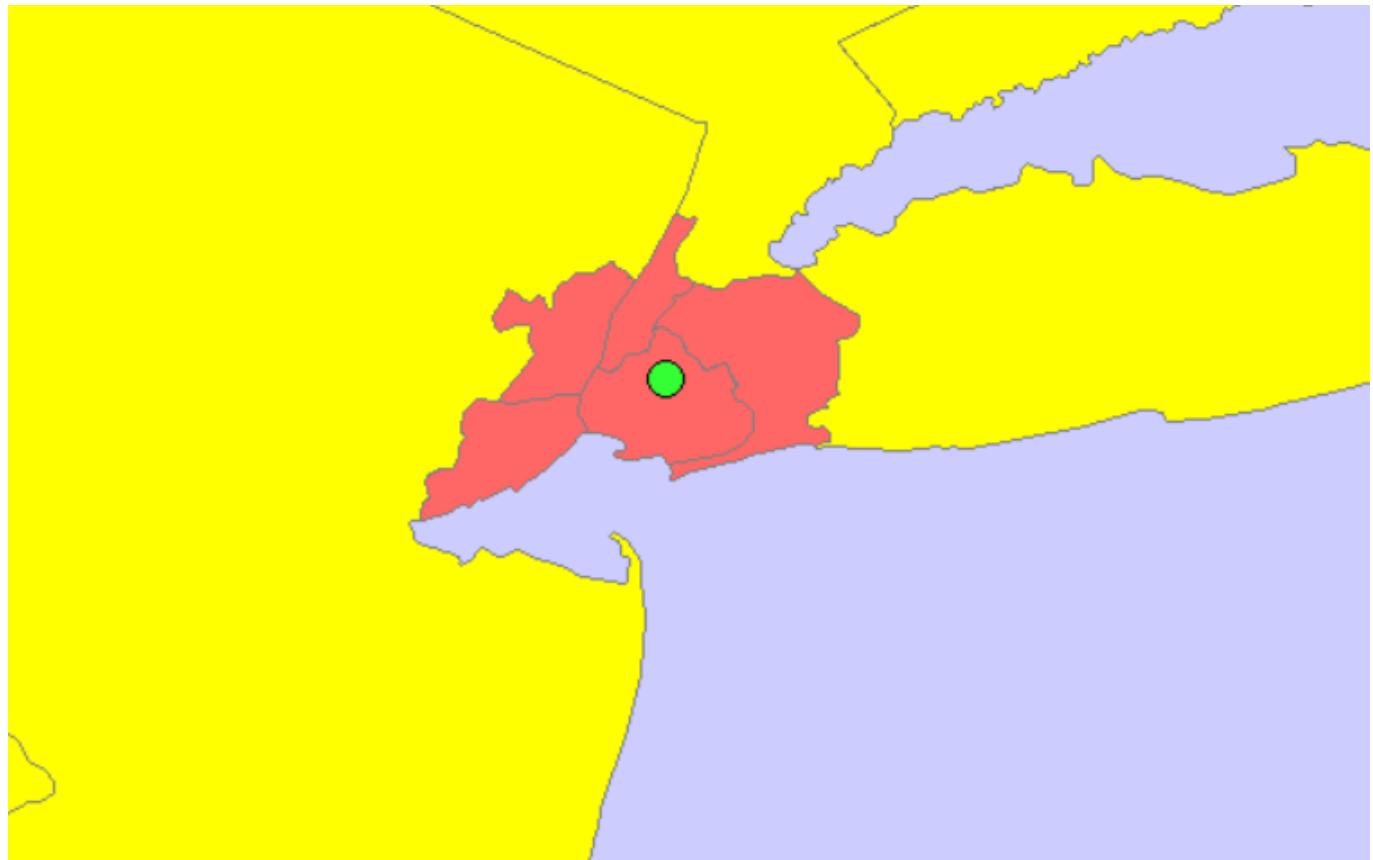
c) Znajdź kilka jednostek administracyjnych (us_counties) z których jest najbliżej do Nowego Jorku

```
select * from us_states
where state_abrv != 'AK';

select * from us_cities
where city = 'New York';

SELECT cn.county, cn.geom
FROM us_counties cn
WHERE cn.id in (
select cn.id
FROM us_counties cn,
us_cities c
WHERE c.city = 'New York'
AND sdo_nn(cn.geom, c.location, 'sdo_num_res=5') = 'TRUE'
);
```

Oczywiście jednostka administracyjna Nowy Jork jest najbliżej miasta Nowy Jork



d) Znajdz 5 najbliższych miast od drogi 'I170', podaj odległość do tych miast

```
select * from us_states
where state_abrv != 'AK';

select * from us_interstates
where interstate = 'I170';

SELECT c.city, c.location
FROM us_cities c
WHERE c.id in (
    select c.id
    FROM us_interstates i, us_cities c
    WHERE i.interstate = 'I170'
    AND sdo_nn(c.location, i.geom, 'sdo_num_res=5') = 'TRUE'
);

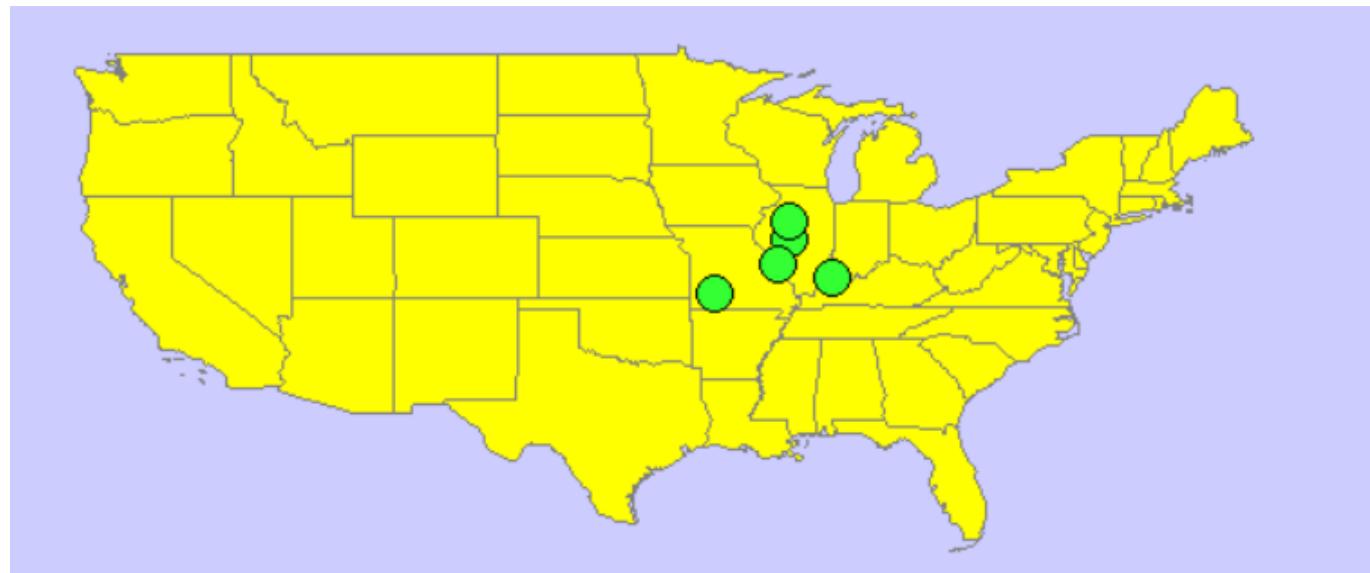
SELECT c.city, c.location,
       SDO_GEM.SDO_DISTANCE(c.location, i.geom, 0.005,'unit=MILE') AS
distance
FROM us_cities c, us_interstates i
WHERE c.id IN (
    SELECT c.id
    FROM us_interstates i, us_cities c
    WHERE i.interstate = 'I170'
    AND SDO_NN(c.location, i.geom, 'sdo_num_res=5') = 'TRUE'
)
```

```
AND i.interstate = 'I170';
)
```

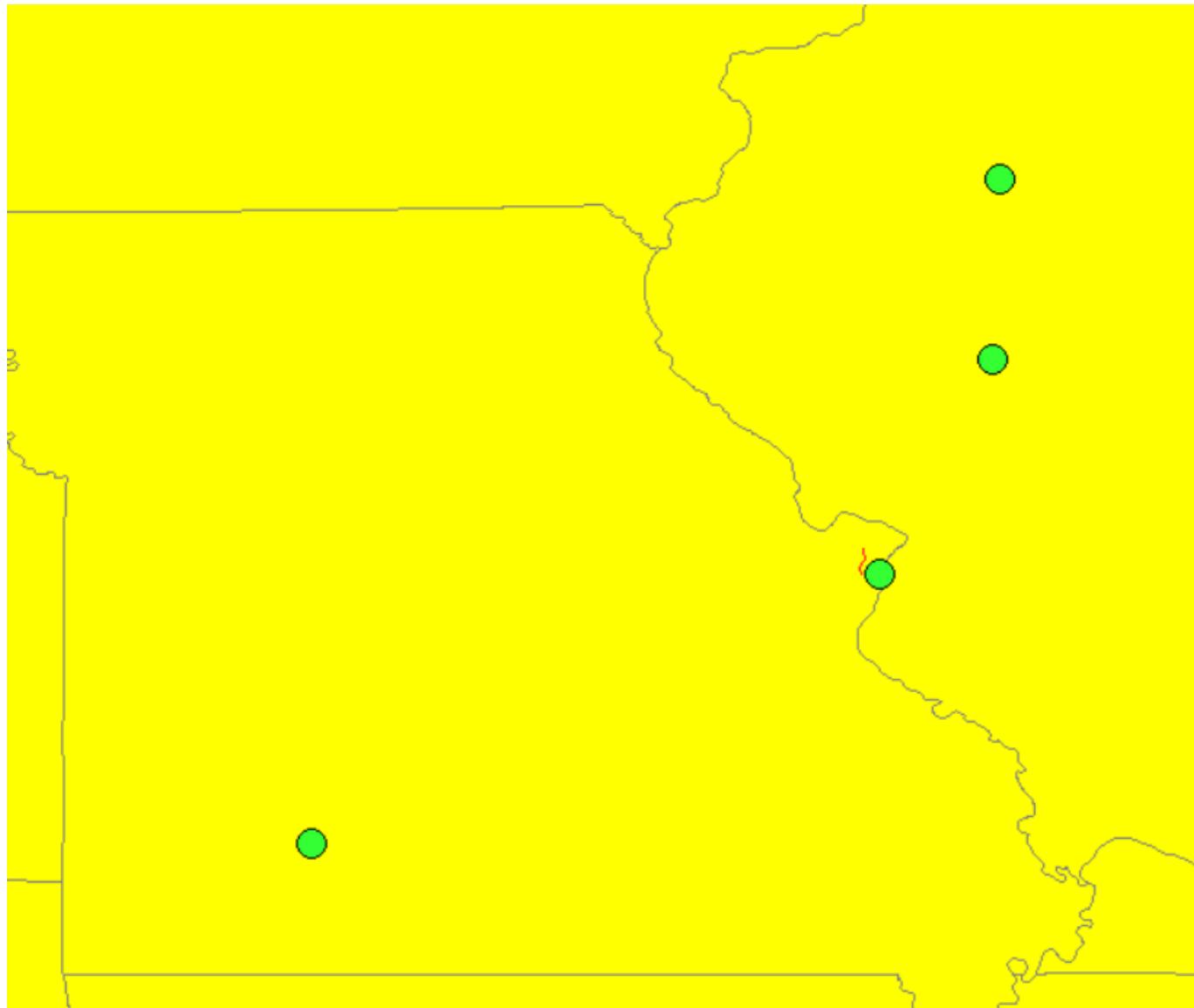
znalezione odległości dla miast to:

city	distance in miles
Evansville	158.224219737852
Peoria	78.7997463714433
Springfield	78.7997463714433
St Louis	5.36297295124004
Springfield	188.508631077882

Znaleźliśmy dwa miasta o nazwie Springfield!



Z początku nie widzieliśmy drogi, jednak po zbliżeniu udało się zobaczyć jej krótki fragment



- e) Znajdz 5 najbliższych dużych miast (o populacji powyżej 300 tys) od drogi 'I170'

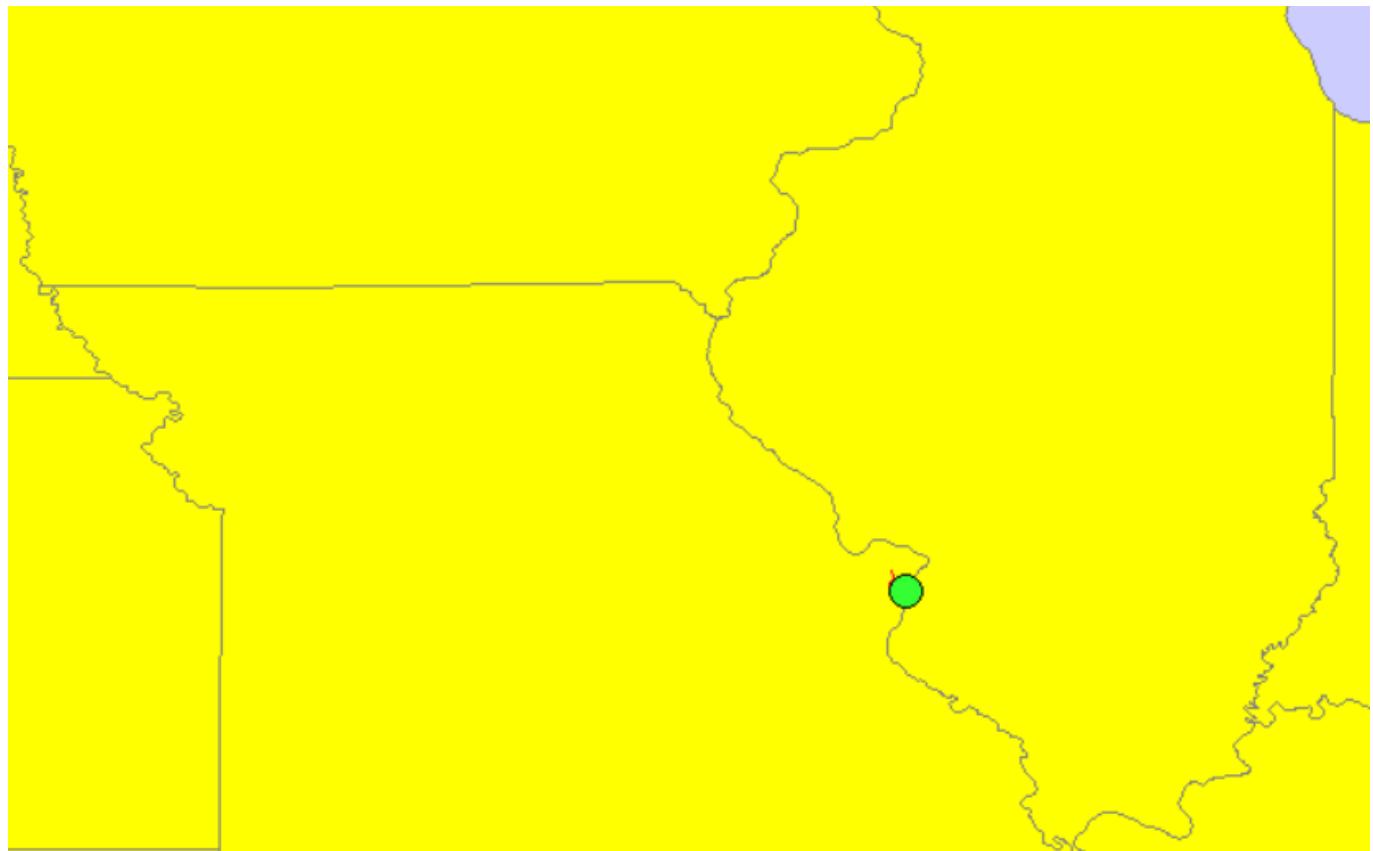
```
select * from us_states
where state_abrv != 'AK';

select * from us_interstates
where interstate = 'I170';

SELECT c2.city, c2.location
FROM us_cities c2
WHERE c2.id in (
    SELECT c.id
    FROM (select * from us_cities where pop90 > 300000) c, us_interstates i
    WHERE i.interstate = 'I170'
    AND SDO_NN(c.location, i.geom, 'sdo_num_res=5') = 'TRUE'
);
```

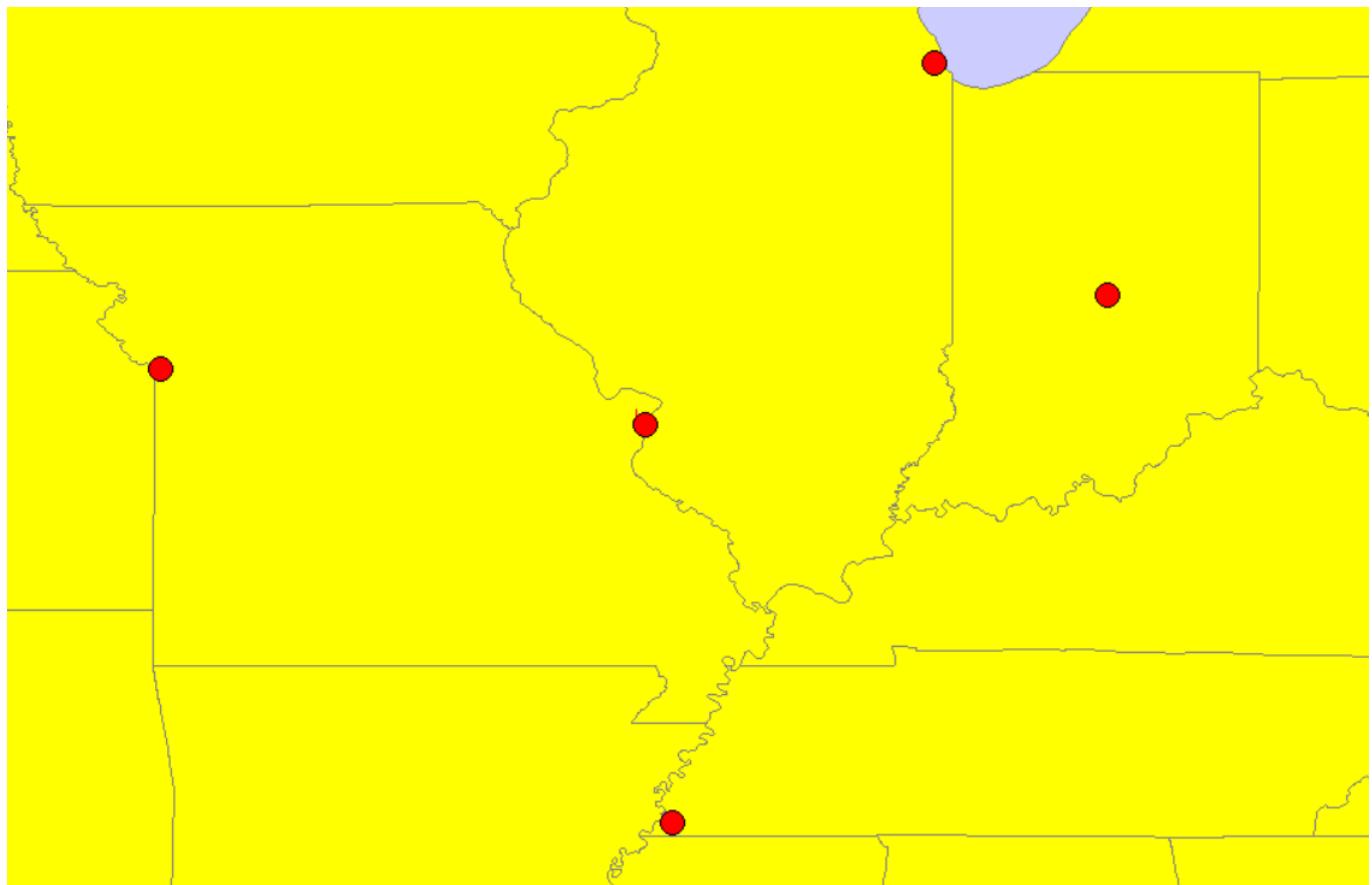
Rozwiążanie nie jest w pełni poprawne. Wygląda na to, że pomimo specyfikowania nowej tabeli c przy pomocy podzapytania **SDO_NN** dalej operuje wyłącznie na tabeli **us_cities** zamiast na efekcie

podzapytania. Zwarcane jest tylko jedno miasto z 5 najbliższych miast - St Louis - jedyne które spełnia warunek o populacji. Zamiast 5 najbliższych spełniających warunek.



Przykład udało się poprawić zmianą podzapytania

```
SELECT c2.city, c2.location
FROM us_cities c2
WHERE c2.id in (
  SELECT c.id
  FROM (select * from us_cities where pop90 > 300000) c, us_interstates i
  WHERE i.interstate = 'I170'
  AND SDO_NN(c.location, i.geom) = 'TRUE'
  AND rownum <= 5
);
```



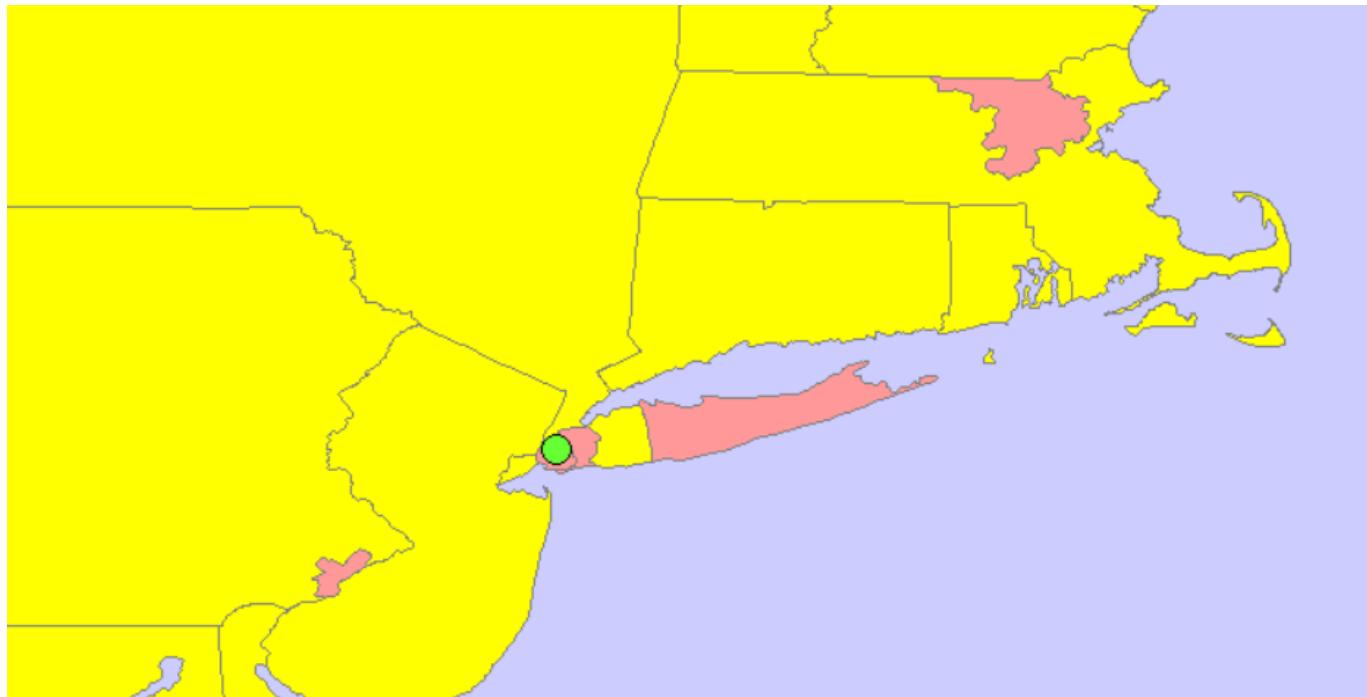
f) 5 jednostek administracyjnych o populacji większej niż 1 300 000 najbliżej Nowego Jorku

```
select * from us_states
where state_abrv != 'AK';

select * from us_cities
where city = 'New York';

SELECT cn.county, cn.totpop, cn.geom
FROM us_counties cn
WHERE cn.id in (
select cn.id
FROM (select * from us_counties where totpop > 1300000 and county != 'New
York') cn, us_cities c
WHERE c.city = 'New York'
AND sdo_nn(cn.geom, c.location) = 'TRUE'
AND rownum <= 5
);
```

W zapytaniu zawarto filtrację która upewnia się, żeby nie został wybrany stan **New York** jednak punkt będący współrzędnymi miasta Nowy Jork znajduje się w jednym ze zwróconych obszarów. Jest to duże miasto które obejmuje kilka obszarów administracyjnych więc punkt określający jego współrzędne znalazł się na obszarze innego hrabstwa



Zadanie 7

Oblicz długość drogi I4

```
SELECT SDO_GEOGRAPHY.SDO_LENGTH (geom, 0.5, 'unit=kilometer') length
FROM us_interstates
WHERE interstate = 'I4';
```

Wyniki, zrzut ekranu, komentarz

The screenshot shows a database management interface with the following components:

- Toolbar:** Includes icons for running queries, saving, opening files, and other database operations.
- Worksheet Tab:** Active tab showing the SQL query:

```
SELECT SDO_GEOGRAPHY.SDO_LENGTH (geom, 0.5, 'unit=kilometer') length
FROM us_interstates
WHERE interstate = 'I4';
```
- Script Output Tab:** Shows the query results in a table format:

LENGTH
1 212.260756199927

- Query Result Tab:** Shows the results of the query execution: "All Rows Fetched: 1 in 0.027 seconds".

Dodatkowo:

- Oblicz długość rzeki Mississippi

```
SELECT SDO_GEOM.SDO_LENGTH (geom, 0.5,'unit=kilometer') length
FROM us_rivers
WHERE name = 'Mississippi';
```

The screenshot shows the Oracle SQL Developer interface. The top window is titled 'dbmanage.lab.ii.agh.edu.pl' and contains a 'Worksheet' tab with the SQL query. The bottom window is titled 'Script Output' and 'Query Result'. It shows the query was executed successfully with one row fetched in 0.033 seconds. The result table has a single column 'LENGTH' with the value 3860.32566492228.

LENGTH
1 3860.32566492228

Długość rzeki Mississippi to 3860 km.

b) Która droga jest najdłuższa/najkrótsza

- droga najdłuższa

```
SELECT interstate, SDO_GEOM.SDO_LENGTH (geom, 0.5,'unit=kilometer') length
FROM us_interstates
ORDER BY length DESC
FETCH FIRST ROW ONLY;
```

dbmanage.lab.ii.agh.edu.pl

Worksheet Query Builder

```
SELECT interstate, SDO_Geom.SDO_LENGTH (geom, 0.5, 'unit=kilometer') length
FROM us_interstates
ORDER BY length DESC
FETCH FIRST ROW ONLY;
```

Script Output Query Result

All Rows Fetched: 1 in 0.547 seconds

	INTERSTATE	LENGTH
1	I90	4290.6462617249

Map View

Spatial 3

Najdłuższa droga ma 4290.64 km i jest nią I90.

- droga najkrótsza

```
SELECT interstate, SDO_GEOM.SDO_LENGTH (geom, 0.5, 'unit=kilometer') length
FROM us_interstates
ORDER BY length
FETCH FIRST ROW ONLY;
```

SOL dbmanage.lab.ii.agh.edu.pl

Worksheet Query Builder

```
SELECT interstate, SDO_GEOM.SDO_LENGTH (geom, 0.5, 'unit=kilometer') length
FROM us_interstates
ORDER BY length
FETCH FIRST ROW ONLY;
```

Script Output Query Result

SQL | All Rows Fetched: 1 in 0.249 seconds

	INTERSTATE	LENGTH
1	I564	0.462140186764249

Map View

Spatial 3

Najkrótsza droga to I564 i ma 0.46 km.

c) Która rzeka jest najdłuższa/najkrótsza

- rzeka najdłuższa

```
SELECT name, SDO_GEOGRAPHYM.SDO_LENGTH (geom, 0.5,'unit=kilometer') length
FROM us_rivers
ORDER BY length DESC
FETCH FIRST ROW ONLY;
```

SOL dbmanage.lab.ii.agh.edu.pl

Worksheet Query Builder

```
SELECT name, SDO_Geom.SDO_LENGTH (geom, 0.5, 'unit=kilometer') length
FROM us_rivers
ORDER BY length DESC
FETCH FIRST ROW ONLY;
```

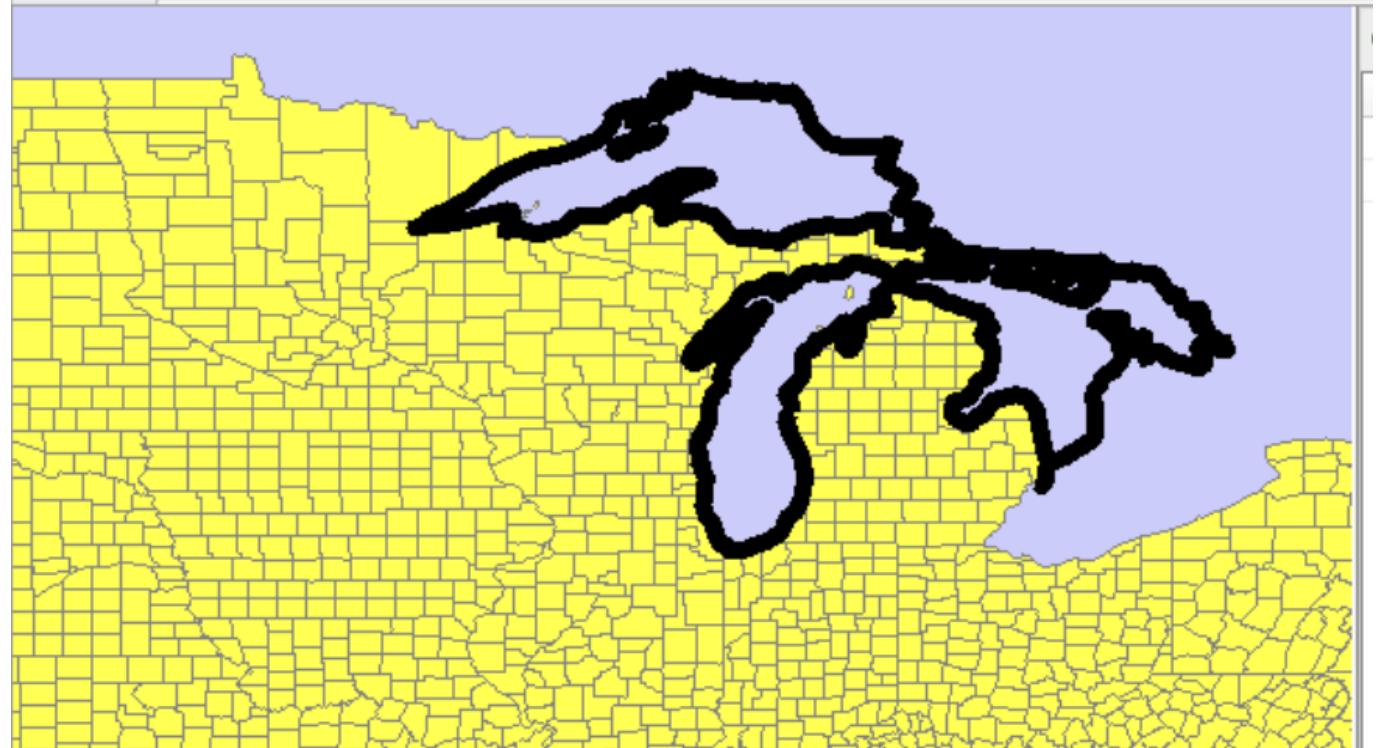
Script Output Query Result

SQL | All Rows Fetched: 1 in 0.048 seconds

	NAME	LENGTH
1	St. Clair	6950.91937515048

Map View

Spatial 3



Najdłuższa rzeka ma długość 6950.91 km i jest nią St. Clair.

- rzeka najkrótsza

```
SELECT name, SDO_GEOM.SDO_LENGTH (geom, 0.5,'unit=kilometer') length
FROM us_rivers
ORDER BY length
FETCH FIRST ROW ONLY;
```

SOL dbmanage.lab.ii.agh.edu.pl x

Worksheet Query Builder

```
SELECT name, SDO_Geom.SDO_LENGTH (geom, 0.5,'unit=kilometer') length
FROM us_rivers
ORDER BY length
FETCH FIRST ROW ONLY;
```

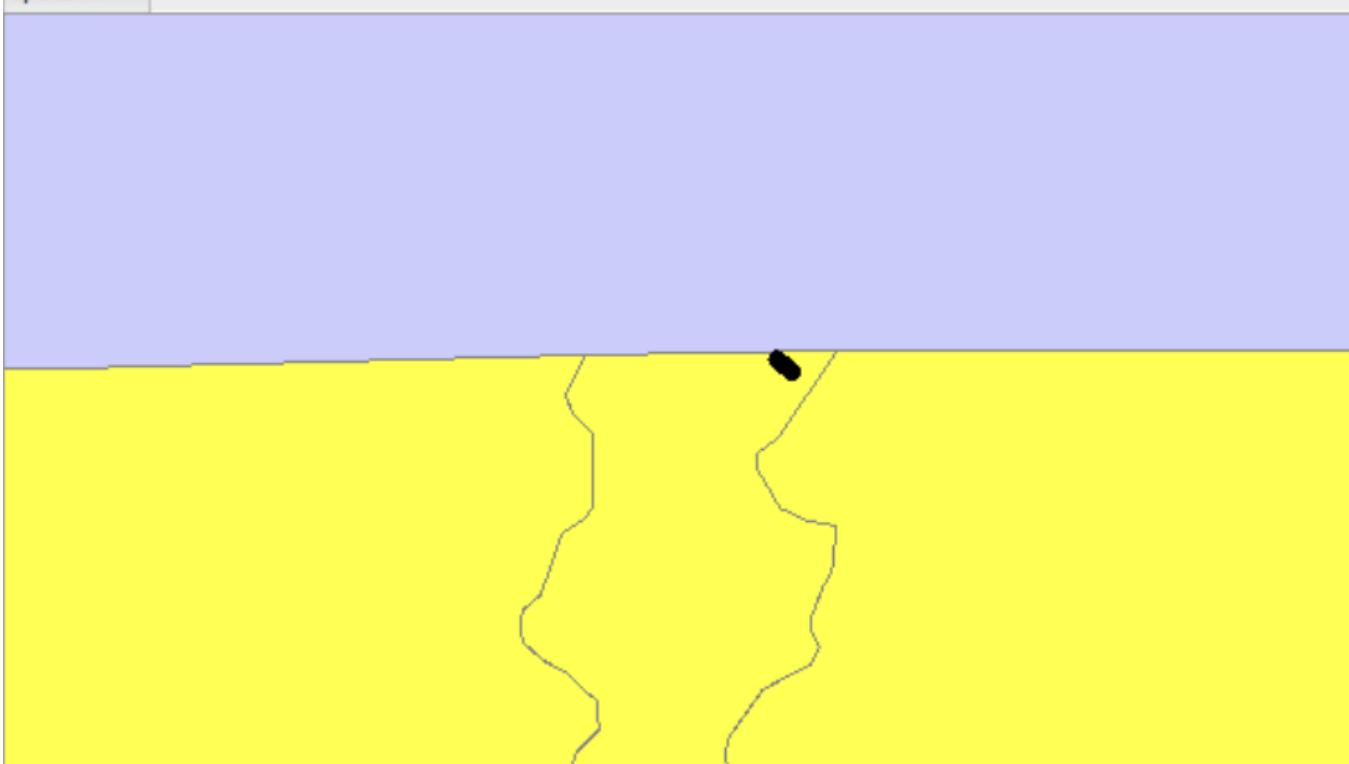
Script Output x | Query Result x

SQL | All Rows Fetched: 1 in 0.039 seconds

	NAME	LENGTH
1	Richelieu	1.16169766454518

Map View x

Spatial 3



Najkrótsza rzeka to Richelieu i ma długość 1.16 km.

Wniosek: W przypadku dróg rezultat był stosunkowo poprawny, natomiast rzeki w używanej bazie danych nie zawsze są przedstawione jako prosta linia, lecz czasami mają bardziej skomplikowany

kształt, przez co otrzymany wynik (długość krzywej) nie do końca zgadza się z długością rzeki podawaną w innych źródłach.

- d) Które stany mają najdłuższą granicę

```
SELECT
    state,
    SDO_GEM.SDO_LENGTH(geom, 0.5, 'unit=kilometer') AS border_length
FROM
    us_states
ORDER BY
    border_length DESC
FETCH FIRST 5 ROWS ONLY
```

dbmanage.lab.ii.agh.edu.pl

Worksheet Query Builder

```
SELECT
    state,
    SDO_Geom.SDO_Length(geom, 0.5, 'unit=kilometer') AS border_length
FROM
    us_states
ORDER BY
    border_length DESC
FETCH FIRST 5 ROWS ONLY
```

Script Output Query Result

All Rows Fetched: 5 in 0.085 seconds

	STATE	BORDER_LENGTH
1	Alaska	26138.3745019651
2	Texas	6779.84795094551
3	California	4145.76647746918
4	Michigan	4140.12257047995
5	Florida	3725.07858238253

Map View

Spatial 3

The screenshot displays a spatial database interface. In the top-left, a code editor window shows an SQL query to select the state, calculate its border length using the SDO_Geom function, and order the results by border length. The top-right shows the execution results in a table, listing the top five states with their names and calculated border lengths. Below the table is a map view window titled 'Spatial 3' which displays the geographic shapes of US states. Alaska is highlighted in light green, while the continental US states are shaded in various tones of yellow and orange, visually representing the border lengths. The map clearly indicates that Alaska has the longest borders among the contiguous states.

Mozemy zauwazyc, ze 3 stany, ktore posiadaja najdluzsze granice to:

- Alaska
- Texas

- California

- e) Itp. (własne przykłady)

Wyniki, zrzut ekranu, komentarz (dla każdego z podpunktów)

- 5 najdłuższych granic (w kilometrach):

```
SELECT cntry_name, SDO_GEOM.SDO_LENGTH(geometry, 0.01, 'unit=kilometer')
as country_length
FROM world_countries
ORDER BY country_length desc;
```

1	Antarctica	(null)
2	Canada	128273,236775619
3	Russia	89169,8793674268
4	United States	64147,0332648669
5	Indonesia	50365,0273828685

- 10 największych parków (w milach):

```
SELECT name, SDO_GEOM.SDO_LENGTH(geom, 0.01, 'unit=mile') as length
FROM us_parks
ORDER BY length desc;
```

	NAME	LENGTH
1	Toiyabe NF	1681,86058696904
2	Humboldt NF	1374,82235485485
3	Lolo NF	1363,07720778012
4	Mackinaw SF	1233,64244902223
5	Glacier Bay NP and NPRES	1226,06967179083
6	Beaverhead NF	1198,14647742855
7	Mark Twain NF	1147,47747648041
8	Kaniksu NF	1102,92704712568
9	Lake Superior SF	1092,1609831483
10	Jefferson NF	1086,84674691652

Oblicz odległość między miastami Buffalo i Syracuse

```
SELECT SDO_GEOGRAPHICAL_DISTANCE ( c1.location, c2.location, 0.5) distance
FROM us_cities c1, us_cities c2
WHERE c1.city = 'Buffalo' and c2.city = 'Syracuse';
```

Wyniki, zrzut ekranu, komentarz

The screenshot shows the Oracle SQL Developer interface with three main panes:

- Worksheet**: Contains the SQL query:

```
SELECT SDO_GEOGRAPHICAL_DISTANCE ( c1.location, c2.location, 0.5, 'unit=kilometer') distance
FROM us_cities c1, us_cities c2
WHERE c1.city = 'Buffalo' and c2.city = 'Syracuse';
```
- Query Result**: Shows the result of the query:

DISTANCE
1 222.184610363969

All Rows Fetched: 1 in 0.079 seconds
- Map View**: A map of the United States showing state boundaries. Two red dots mark the locations of Buffalo and Syracuse. The distance between them is indicated by a purple line.

Odległość między miastami Buffalo i Syracuse wynosi 222.18 km.

Dodatkowo:

- a) Oblicz odległość między miastem Tampa a drogą I4

```
SELECT SDO_GEOGRAPHICAL_DISTANCE ( c1.location, i.geom, 0.5, 'unit=kilometer')
distance
```

```
FROM us_cities c1, us_interstates i
WHERE c1.city = 'Tampa' and i.interstate = 'I4';
```

Screenshot of the dbmanange.lab.ii.agh.edu.pl application interface showing a spatial query and its results.

The top section shows the SQL query:

```
SELECT SDO_Geom.SDO_DISTANCE (c1.location, i.geom, 0.5, 'unit=kilometer') distance
FROM us_cities c1, us_interstates i
WHERE c1.city = 'Tampa' and i.interstate = 'I4';
```

The middle section displays the results in a table:

DISTANCE
1 3.10391172130556

The bottom section is a Map View showing a red dot representing Tampa and a black line representing Interstate I4. The distance between them is indicated by a small line segment.

Odległość między miastem Tampa a drogą I4 to 3.10 km.

- b) Jaka jest odległość między stanem Nowy Jork a Florydą

```
SELECT SDO_GEM.SDO_DISTANCE (s1.geom, s2.geom, 0.5, 'unit=kilometer')
distance
FROM us_states s1, us_states s2
WHERE s1.state = 'New York' and s2.state = 'Florida';
```

dbmanage.lab.ii.agh.edu.pl

Worksheet Query Builder

```
SELECT SDO_GEOGRAPHY.SDO_DISTANCE (s1.geom, s2.geom, 0.5, 'unit=kilometer') distance
FROM us_states s1, us_states s2
WHERE s1.state = 'New York' and s2.state = 'Florida';
```

Script Output Query Result

SQL | All Rows Fetched: 1 in 0.037 seconds

DISTANCE
1 1256.58387785727

Map View

Spatial 3

Odległość między stanem Nowy Jork a Florydą wynosi 1256.58 km.

- c) Jaka jest odległość między miastem Nowy Jork a Florydą

```
SELECT SDO_GEOM.SDO_DISTANCE (c1.location, s2.geom, 0.5, 'unit=kilometer')
distance
FROM us_cities c1, us_states s2
WHERE c1.city = 'New York' and s2.state = 'Florida';
```

SOL dbmanage.lab.ii.agh.edu.pl

Worksheet Query Builder

```
SELECT SDO_GEOM.SDO_DISTANCE (c1.location, s2.geom, 0.5, 'unit=kilometer') distance
FROM us_cities c1, us_states s2
WHERE c1.city = 'New York' and s2.state = 'Florida';
```

Script Output x | Query Result x

SQL | All Rows Fetched: 1 in 0.027 seconds

DISTANCE
1 1296.59076150732

Map View x

Spatial 3

A screenshot of a spatial database management interface. The top section shows a SQL query in the 'Worksheet' tab, which calculates the distance between New York City and Florida. The result is displayed in the 'Query Result' tab, showing a single row with a value of 1296.59076150732. The bottom section, titled 'Spatial 3', contains a map view where the state of New York is highlighted in yellow and a red dot marks a point on its southern coast. Below it, the state of Florida is also highlighted in yellow.

Odległość między miastem Nowy Jork a Florydą to 1296.59 km.

- d) Podaj 3 parki narodowe do których jest najbliżej z Nowego Jorku, oblicz odległości do tych parków

```
SELECT p.name,
       SDO_GEOM.SDO_DISTANCE(c.location, p.geom, 0.5, 'unit=mile') AS
distance
FROM us_parks p,
     us_cities c
WHERE c.city = 'New York'
AND SDO_NN(p.geom, c.location, 'sdo_num_res=3') = 'TRUE'
ORDER BY distance
```

dbmanage.lab.ii.agh.edu.pl

Worksheet Query Builder

```
SELECT
    p.name,
    SDO_GEOGRAPHICAL_DISTANCE(c.location, p.geom, 0.5, 'unit=mile') AS distance
FROM
    us_parks p,
    us_cities c
WHERE
    c.city = 'New York'
```

Script Output Query Result | All Rows Fetched: 3 in 0.079 seconds

	NAME	DISTANCE
1	Institute Park	0.956845723074769
2	Prospect Park	1.06755874464742
3	Thompkins Park	1.32697342712507

Map View

Spatial 3

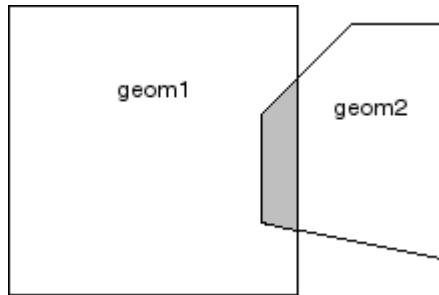
Trzy najbliższe parki to Institute Park, Prospect Park oraz Thompkins Park.

- e) Przetestuj działanie funkcji

a. `sdo_intersection, sdo_union, sdo_difference`

sdo_intersection

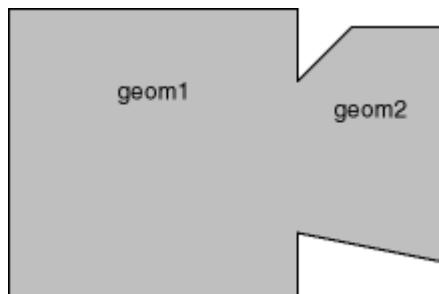
```
SELECT SDO_GEOG.SDO_INTERSECTION(a.geom, b.geom, 0.005) AS  
intersection_geom  
FROM geometry_table a, geometry_table b  
WHERE a.id = 1 AND b.id = 2;
```



Wniosek: Funkcja SDO_INTERSECTION zwraca część wspólną dwóch podanych geometrii. Zwraca nowy obiekt geometryczny, który reprezentuje wspólną część obu wejściowych geometrii.

sdo_union

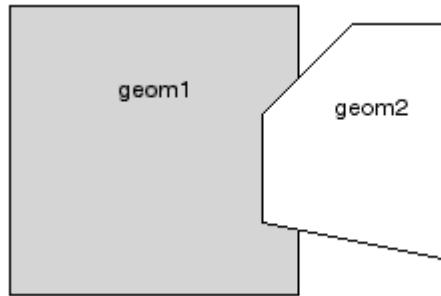
```
SELECT SDO_GEOG.SDO_UNION(a.geom, b.geom, 0.005) AS union_geom  
FROM geometry_table a, geometry_table b  
WHERE a.id = 1 AND b.id = 2;
```



Wniosek: Funkcja SDO_UNION zwraca sumę dwóch podanych geometrii. Zwraca nowy obiekt geometryczny, który obejmuje obszar obu wejściowych geometrii.

sdo_difference

```
SELECT SDO_GEOG.SDO_DIFFERENCE(a.geom, b.geom, 0.005) AS difference_geom  
FROM geometry_table a, geometry_table b  
WHERE a.id = 1 AND b.id = 2;
```



Wniosek: Funkcja SDO_DIFFERENCE różnice między dwoma podanymi geometriami. Zwraca nowy obiekt geometryczny, który reprezentuje obszar pierwszej geometrii, z którego usunięto obszar drugiej geometrii.

b. sdo_buffer

sdo_buffer

```
SELECT SDO_GEOGRAPHICAL.MSDO_BUFFER(geom, 0.1, 0.005) AS buffer_geom  
FROM geometry_table  
WHERE id = 1;
```

Wniosek: Funkcja SDO_BUFFER tworzy bufer wokół podanej geometrii. Zwraca nową geometrię, która reprezentuje obszar w określonej odległości od wejściowej geometrii.

c. sdo_centroid, sdo_mbr, sdo_convexhull, sdo_simplify

sdo_centroid

```
SELECT SDO_GEOGRAPHICAL.MSDO_CENTROID(geom, 0.005) AS centroid_geom  
FROM geometry_table  
WHERE id = 1;
```

Wniosek: Funkcja SDO_CENTROID oblicza centroid dla podanej geometrii. Zwraca punkt geometryczny, który jest środkiem ciężkości wejściowej geometrii.

sdo_mbr

```
SELECT SDO_GEOGRAPHICAL.MSDO_MBR(geom) AS mbr_geom  
FROM geometry_table  
WHERE id = 1;
```

Wniosek: Funkcja SDO_MBR zwraca minimalny zewnętrzny prostokąt dla podanej geometrii. Zwraca prostokątną geometrię, która całkowicie obejmuje wejściowy obiekt geometryczny.

sdo_convexhull

```
SELECT SDO_GEOGRAPHY.MBR_TO_GEOGRAPHY(SDO_CONVEXHULL(geom, 0.005)) AS convexhull_geom
FROM geometry_table
WHERE id = 1;
```

Wniosek: Funkcja SDO_CONVEXHULL tworzy otoczkę wypukłą wokół podanej geometrii. Zwraca najmniejszy wypukły wielokąt, który obejmuje wszystkie punkty wejściowej geometrii.

sdo_simplify

```
SELECT SDO_GEOGRAPHY.MBR_TO_GEOGRAPHY(SDO_SIMPLIFY(geom, 0.005)) AS simplified_geom
FROM geometry_table
WHERE id = 1;
```

Wniosek: Funkcja SDO_SIMPLIFY upraszcza podaną geometrię. Zwraca uproszoną wersję wejściowej geometrii, zachowującą jej ogólny kształt i cechy, ale z mniejszą liczbą punktów.

f) Itp. (własne przykłady)

Wyniki, zrzut ekranu, komentarz (dla każdego z podpunktów)

- Pola powierzchni stanów od największego do najmniejszego w kilometrach kwadratowych (5 przykładów)

```
SELECT state, SDO_GEOGRAPHY.MBR_TO_GEOGRAPHY(SDO_AREA(geom, 0.5, 'unit=SQ_KM')) AS area
FROM us_states
ORDER BY area DESC
```

('Alaska', 1501593.16) ('Texas', 687005.67) ('California', 410033.45) ('Montana', 380815.66) ('New Mexico', 314906.25)

e) Wykresy na mapie USA

```
ALTER SESSION SET CURRENT_SCHEMA = US_SPAT;

select * from us_states
where state_abrv != 'AK';

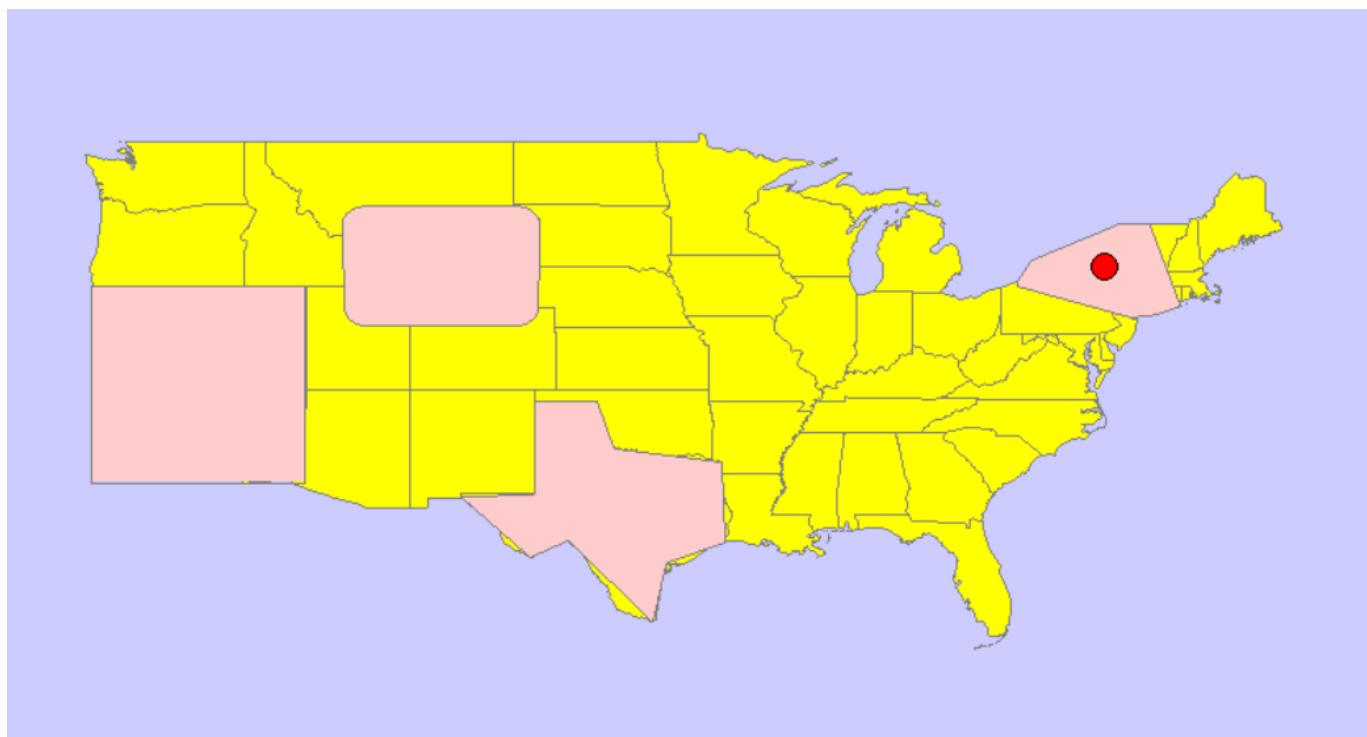
SELECT sdo_geom.sdo_buffer(c.geom, 100000, 0.5) AS buffer_geom
FROM us_states c
WHERE c.state = 'Wyoming';
```

```
SELECT sdo_geom.sdo_centroid(c.geom, 0.5) as centroid
FROM us_states c
WHERE c.state = 'New York';

SELECT sdo_geom.sdo_convexhull(c.geom) as hull
FROM us_states c WHERE c.state = 'New York';

SELECT sdo_util.simplify(c.geom, 100000) as simplify
FROM us_states c WHERE c.state = 'Texas';

SELECT sdo_geom.sdo_mbr(c.geom) as mbr
FROM us_states c WHERE c.state = 'California';
```



Zadanie 8

Wykonaj kilka własnych przykładów/analiz

Wyniki, zrzut ekranu, komentarz

- Analiza przykładu najdłuższej rzeki przy użyciu narzędzi dostarczanych przez język Python (folium, geojson)

Skrypt w Pythonie

```
# Initialize an empty folium map
m = folium.Map()

# Define the SQL query to find the longest river and counties that
# intersect with it
query = """
```

```
WITH LongestRiver AS (
  SELECT r.geom
  FROM us_rivers as r
  ORDER BY SDO_GEOM.SDO_LENGTH(r.geom, 0.005) DESC
  FETCH FIRST 1 ROWS ONLY)

  SELECT sdo_util.to_wktgeometry(c.geom)
  FROM us_counties as c, LongestRiver as lr
  WHERE SDO_ANYINTERACT (c.geom, lr.geom) =
  'TRUE'
  .....

# Execute the SQL query and fetch all results
# This returns the geometries of counties that intersect with the longest
river in WKT format
results = cursor.execute(query).fetchall()

# Define the style for the GeoJSON features
style = {"fillColor": "blue", "color": "red"}

l = []
for row in results:
    # Create a GeoJSON feature with the geometry and empty properties and
add to list
    g = geojson.Feature(geometry=row[0], properties={})
    l.append(g)

# Create a GeoJSON FeatureCollection from the list of features
feature_collection = geojson.FeatureCollection(l)

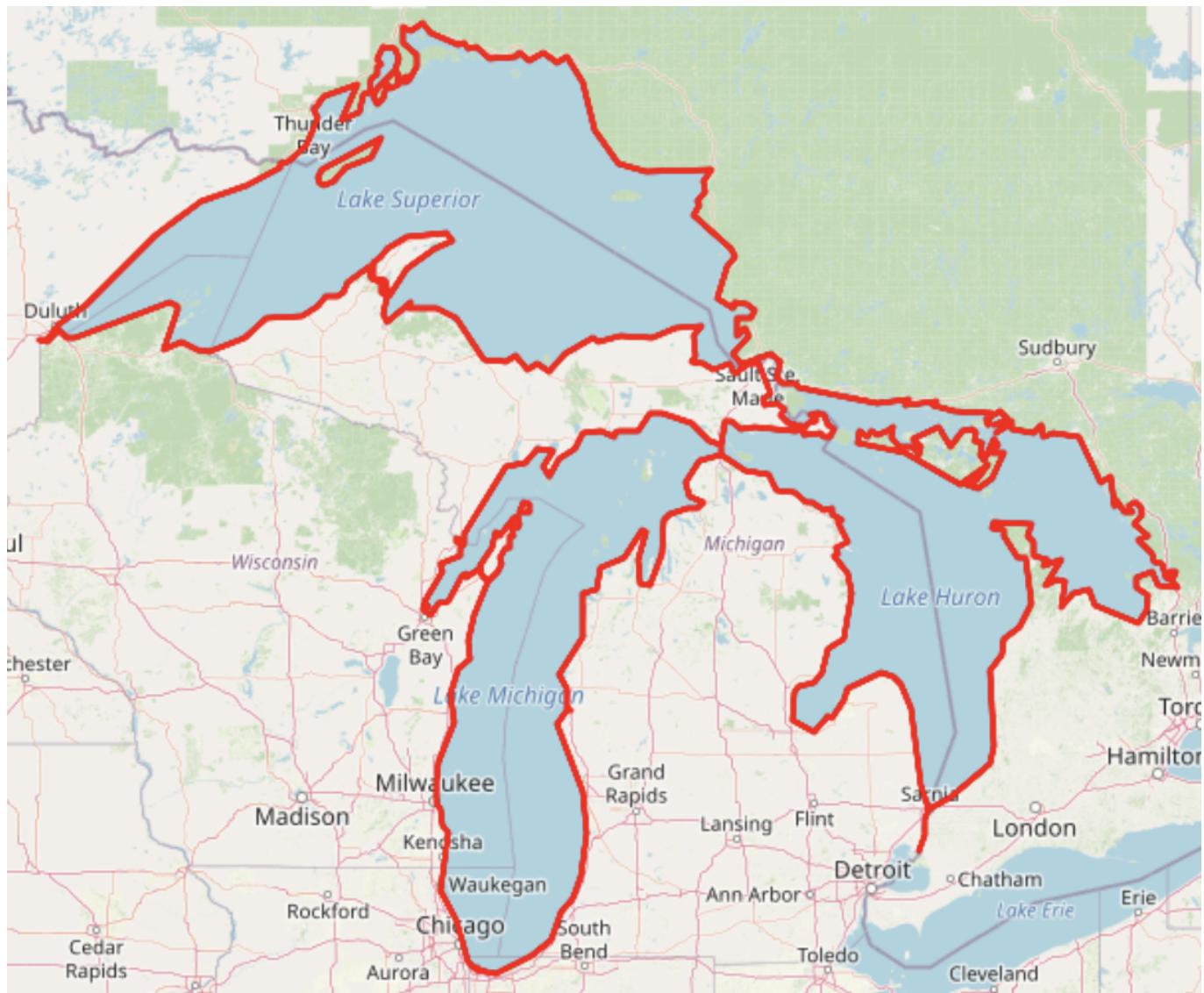
# Add the GeoJSON data to the folium map with the specified style
folium.GeoJson(feature_collection, style_function=lambda x:
style).add_to(m)
```

Wynik



Wniosek

Wynik uzyskany przy pomocy języka Python jest taki sam jak przy pomocy narzędzi postgresa. W obydwu jednak przypadkach, możemy zauważyc, że kształt który jest przedstawiony nie wygląda jak rzeka a bardziej jezioro. Mamy ewidentnie do czynienia z błędem w danych. Po sprawdzeniu wygląda na to, że rzeka St. Clair to niewielka kreska w okolicach Detroit. Możemy to lepiej zaobserwować na przybliżonym zdjęciu:



2. Stany w postaci minimalnych pokrywajacych sie prostokatow

Skrypt w Pythonie

```
# Initialize map
m = folium.Map()

# New query to get the MBR of each state geometry
query = """
SELECT SDO_UTIL.T0_WKTGEOMETRY(SDO_GEOEM.SDO_MBR(c.geom))
FROM us_states c
"""

# Execute the query and fetch results
results = cursor.execute(query).fetchall()

# Define style for GeoJson
style = {"fillColor": "blue", "color": "red"}

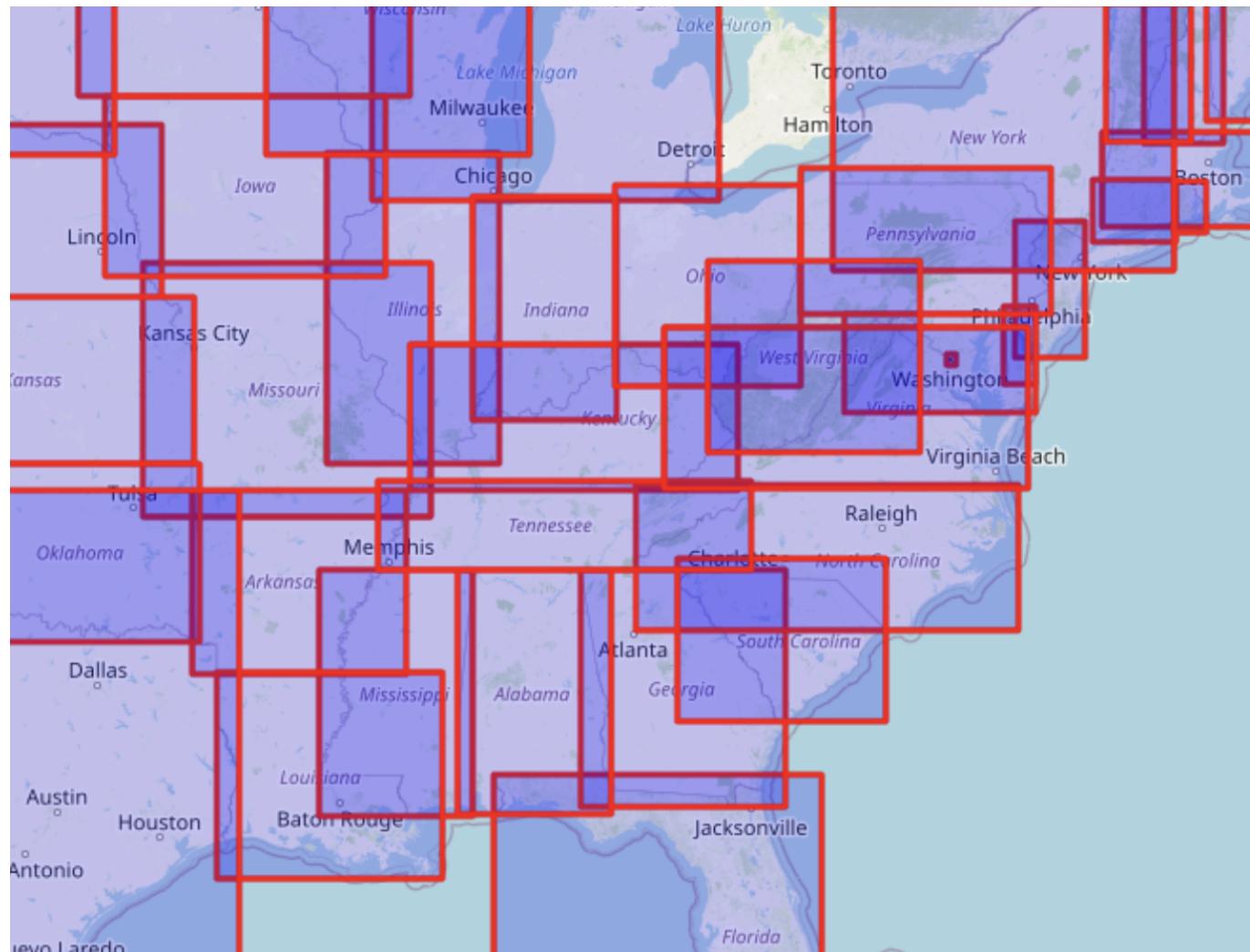
# Process the query results into GeoJSON features
features = []
for row in results:
```

```
# Convert WKT to GeoJSON geometry
geom = wkt.loads(row[0])
geojson_geom = geojson.Feature(geometry=shape(geom), properties={})
features.append(geojson_geom)

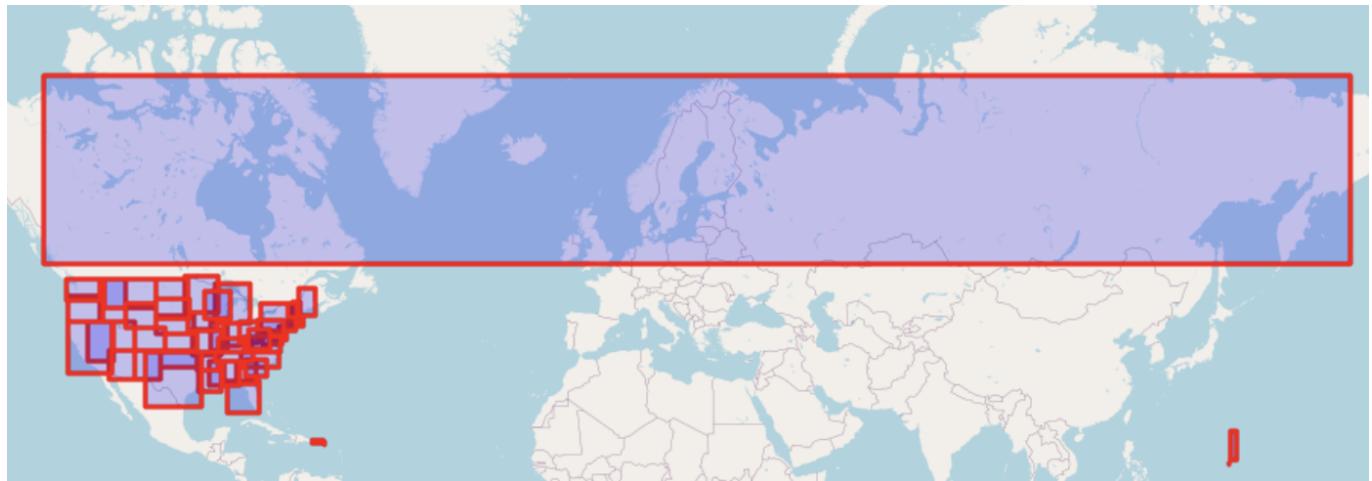
# Create a GeoJSON FeatureCollection
feature_collection = geojson.FeatureCollection(features)

# Add the GeoJSON data to the map with styling
folium.GeoJson(feature_collection, style_function=lambda x:
style).add_to(m)
```

Wynik



Wniosek: W tym przypadku, udało się zaobserwować pewną anomalię ze stanem Alaska. Mozemy zaobserwować, że prostokąt który jest powiązany z Alaską, nawet jej nie dotycza.



3. Najdłusza rzeka w przepływająca przez stan Texas:

Skrypt w Pythonie

```
# Initialize the map centered on Texas
m = folium.Map(location=[31.0, -100.0], zoom_start=6)

# Query to get the longest river in Texas
query_longest_river_in_texas = """
SELECT r.name, SDO_GEM.SDO_LENGTH(r.geom, 0.005, 'unit=KM') as length,
SDO_UTIL.TO_WKTGEOMETRY(r.geom)
FROM us_rivers r, us_states s
WHERE s.state = 'Texas' AND SDO_ANYINTERACT(r.geom, s.geom) = 'TRUE'
ORDER BY length DESC
FETCH FIRST ROW ONLY
"""

# Execute the query and fetch the result
result = cursor.execute(query_longest_river_in_texas).fetchone()

# Extract river name, length, and geometry
river_name = result[0]
river_length = result[1]
river_geom_wkt = result[2]

# Convert WKT to GeoJSON geometry
river_geom = wkt.loads(river_geom_wkt)
river_geojson_geom = geojson.Feature(
    geometry=shape(river_geom),
    properties={"name": river_name, "length_km": river_length},
)

# Define style for the river
style = {"fillColor": "blue", "color": "red"}

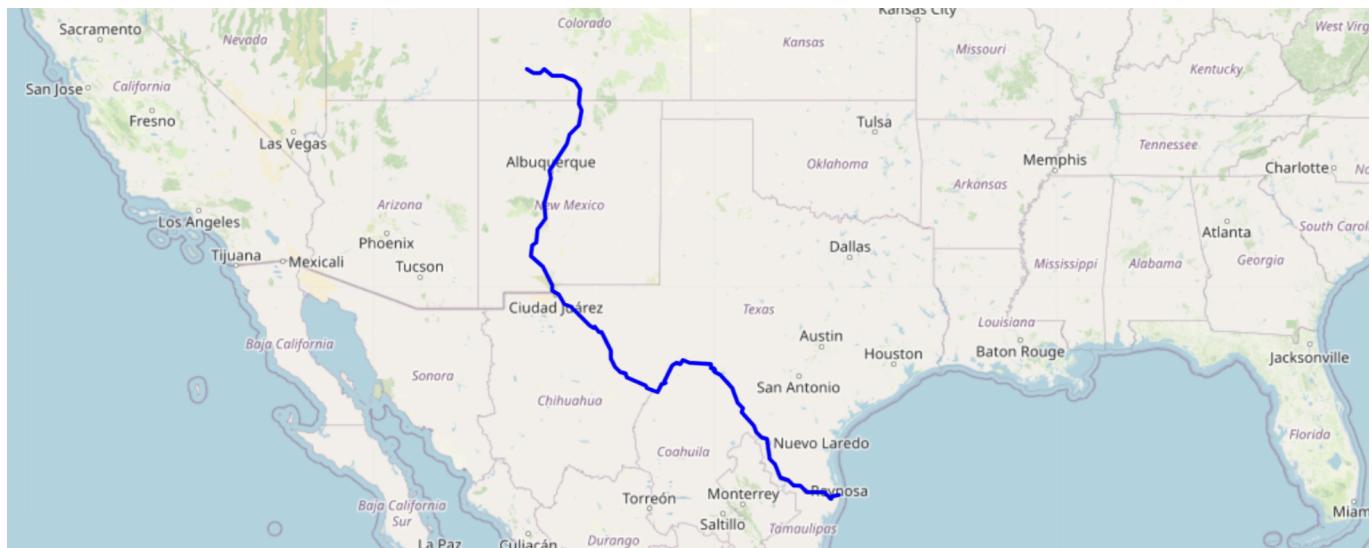
# Create a GeoJSON FeatureCollection
feature_collection = geojson.FeatureCollection([river_geojson_geom])

# Add the GeoJSON data to the map with styling
```

```
folium.GeoJson(feature_collection, style_function=lambda x:
style).add_to(m)

# Add a popup with the river name and length
popup_content = f"<strong>{river_name}</strong><br>Length: {river_length} km"
folium.Popup(popup_content).add_to(m)
```

Wynik



Wniosek: W tym przykładzie podajemy zapytanie, które zwraca i umożliwia wyświetlenie najdłuzszej rzeki przepływającą przez stan Texas (Rio Grande, 2397.13 km).

4. 5 parków narodowych najbliżej Los Angeles

Skrypt w Pythonie

```
# Query to get the nearest parks to Los Angeles
query_nearest_parks_to_la = """
SELECT p.name, SDO_GEM.SDO_DISTANCE(p.geom, c.location, 0.005, 'unit=KM')
as distance, SDO_UTIL.TO_WKTGEOMETRY(p.geom)
FROM us_parks p, us_cities c
WHERE c.city = 'Los Angeles'
ORDER BY distance
FETCH FIRST 5 ROWS ONLY
"""

# Execute the query and fetch the results
results = cursor.execute(query_nearest_parks_to_la).fetchall()

# Initialize the map centered on Los Angeles
m = folium.Map(location=[34.0522, -118.2437], zoom_start=12)

# Define style for the parks
style = {"fillColor": "green", "color": "darkgreen"}
```

```
# Initialize an empty list to store GeoJSON features
features = []

# Process the query results
for row in results:
    park_name = row[0]
    park_distance = row[1]
    park_geom_wkt = row[2]

    # Convert WKT to Shapely geometry
    park_geom = loads(park_geom_wkt)

    # Create a GeoJSON feature with the park's geometry and properties
    park_geojson_geom = geojson.Feature(
        geometry=park_geom, properties={"name": park_name, "distance_km": park_distance}
    )

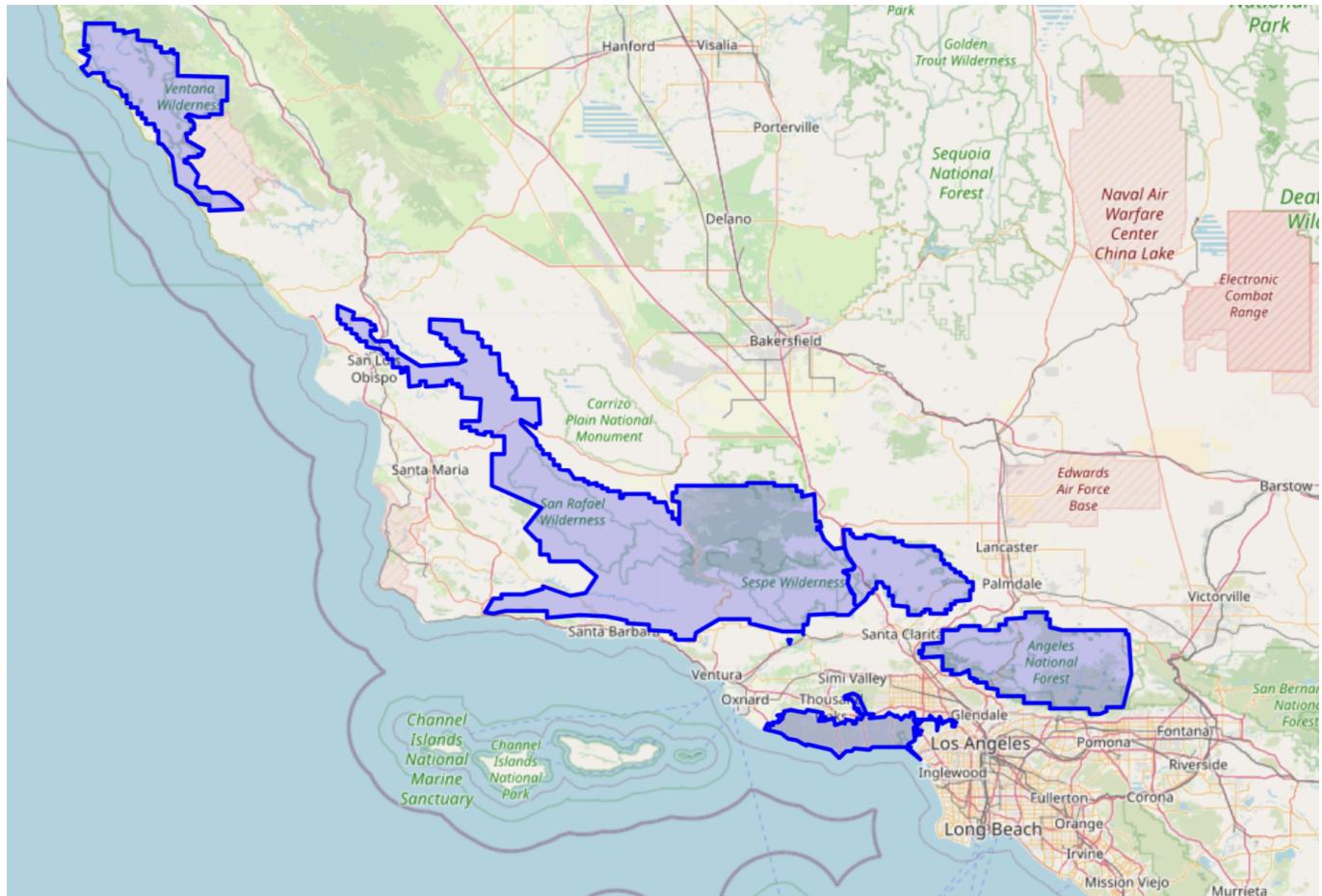
    # Add the feature to the list
    features.append(park_geojson_geom)

    # Add a marker to the map for each park with a popup showing its name
    # and distance
    folium.Marker(
        location=[park_geom.centroid.y, park_geom.centroid.x],
        popup=f"<strong>{park_name}</strong><br>Distance:<br>{park_distance:.2f} km",
        icon=folium.Icon(color="green"),
    ).add_to(m)

# Create a GeoJSON FeatureCollection
feature_collection = geojson.FeatureCollection(features)

# Add the GeoJSON data to the map with styling
folium.GeoJson(feature_collection, style_function=lambda x: style).add_to(m)
```

Wynik



Wniosek: Zapytanie to zwraca i umoliwia nam wyświetlenie na mapie 5 parków narodowych najbliższych Los Angeles (Santa Monica, Mountains NRA, Angeles NF, Los Padres NF, Kenney Grove Park, Toland Park), bez wykorzystania funkcji **SDO_NN**.

Punktacja

zad	pkt
1	0,5
2	1
3	1
4	1
5	3
6	3
7	6
8	4
razem	20