Project Course - Part Two: Querying

Done by: Axel Facundo Preiti Tasat

Query 01

For each vehicle write an SQL query to find the maximum length (in Km) over all its trips. The result should show three columns: vehicleId, time duration of the trip and the trajectory of this trip.

Take into consideration that if a vehicle has multiple trips with the same maximum length, it must appear multiple times in the result set (one for each duration time).

Solution

```
SELECT t.vehicleid, duration(t.trip), t.trajectory, st_length(t.trajectory) AS length
FROM trips t
WHERE st_length(t.trajectory) = (
   SELECT max(st_length(t2.trajectory))
   FROM trips t2
   WHERE t2.vehicleid = t.vehicleid
)
ORDER BY t.vehicleid, length DESC;
```

Query 02

Write an SQL query that computes the trips that cross at least two adjacent municipalities. You should only show the tripid and trips restricted by these boundaries and the time when this intersection occurred.

Limit the result to the first 30 tuples, showing first those with less starting time of the trip. Also display the result in QGIS.

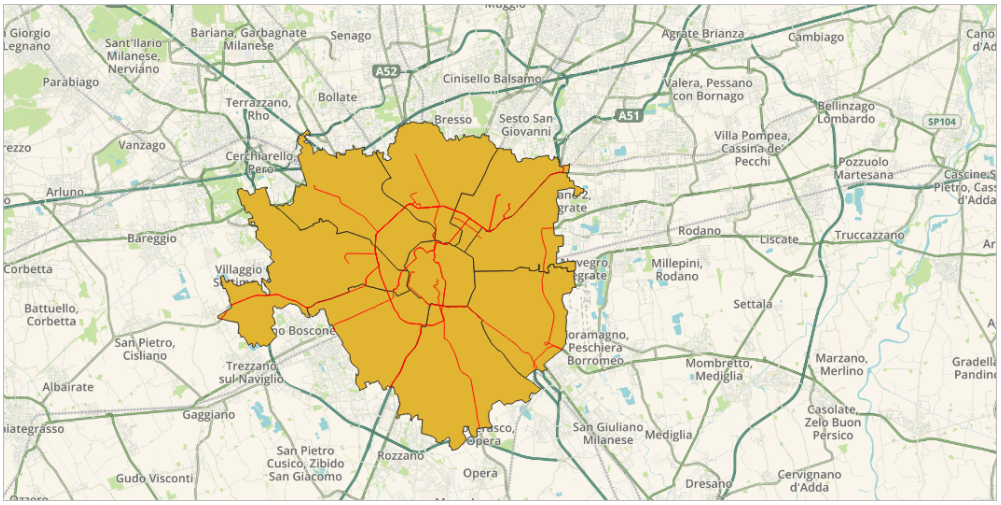
Solution

```
WITH adjacent_municipalities AS (
   SELECT m1.municipalityid AS m1_id,
        m2.municipalityid AS m2_id,
        st_union(m1.municipalitygeo, m2.municipalitygeo) as union_geo,
        st_intersection(m1.municipalitygeo, m2.municipalitygeo) AS boundary_geo
   FROM municipalities m1, municipalities m2
   WHERE st_touches(m1.municipalitygeo, m2.municipalitygeo)
      AND m1.municipalityid < m2.municipalityid
)
SELECT
   t.tripid,
```

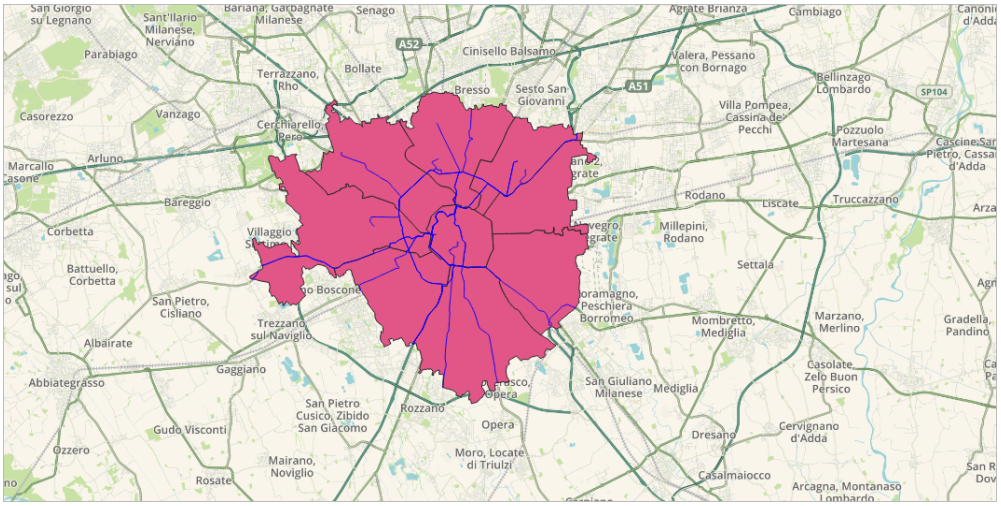
```
    atgeometry(t.trip, am.union_geo) AS trip,
    trajectory(atgeometry(t.trip, am.union_geo)) AS trajectory,
    starttimestamp(atgeometry(t.trip, am.boundary_geo)) AS intersection_time
FROM trips t, adjacent_municipalities am
WHERE eintersects(t.trip, am.boundary_geo)
ORDER BY starttimestamp(t.trip)
LIMIT 30;
```

QGIS Output

Eco-friendly variant



Car-priority variant



Query 03

📋 Write an SQL query to compute when and where a pair of vehicles have been at 50 m or less from each other. If necessary, display only the first 20 pairs of vehicles that satisfy this condition.

Solution

```
WITH near_vehicles AS (
   SELECT DISTINCT ON (t1.vehicleid, t2.vehicleid)
        t1.vehicleid AS vehicle1,
        t2.vehicleid AS vehicle2,
        t1.trip as trip1,
        t2.trip as trip2,
        starttimestamp(atmin(t1.trip <-> t2.trip)) AS time_nearest
   FROM trips t1, trips t2
   WHERE t1.vehicleid < t2.vehicleid AND t1.trip |=| t2.trip <= 50
   ORDER BY t1.vehicleid, t2.vehicleid, t1.trip <-> t2.trip
   LIMIT 20
)
SELECT nv.vehicle1,
      nv.vehicle2,
      nv.time_nearest,
      attime(nv.trip1, nv.time_nearest) AS trip1_location,
      attime(nv.trip2, nv.time_nearest) AS trip2_location
FROM near_vehicles nv;
```

Query 04

Write an SQL query that computes the interval *[t1, t2]* where *t1* and *t2* are the starting and ending times of the whole period generated with the `berlinmod_generate` method. Also compute *t*, the middle point of this interval.

We want to classify trips according to the following criteria:

a. **"the first half of trips"**: those ones that occurred completely inside the interval *[t1, t)*

b. **"the second half of trips"**: those ones that occurred completely inside the interval *[t, t2]*

c. **"trips that crossed t"**: those ones that started in *[t1, t)* and finished in *[t, t2]*.

Propose a query that returns only just one tuple with three columns indicating how many trips belong to each of the categories above

Solution

```
WITH t1 AS (
        SELECT min(starttimestamp(trip)) AS t1
        FROM trips
    ),
   t2 AS (
      SELECT max(endtimestamp(trip)) AS t2
      FROM trips
   ),
   t AS (
      SELECT t1 + (t2 - t1) / 2 AS t
      FROM t1, t2
   ),
   first_half AS (
      SELECT count(*) AS first_half_count
```

```
        FROM trips
        WHERE starttimestamp(trip) >= (SELECT t1 FROM t1)
          AND endtimestamp(trip) < (SELECT t FROM t)
    ),
    second_half AS (
        SELECT count(*) AS second_half_count
        FROM trips
        WHERE starttimestamp(trip) >= (SELECT t FROM t)
          AND endtimestamp(trip) <= (SELECT t2 FROM t2)
    ),
    crossed_t AS (
        SELECT count(*) AS crossed_t_count
        FROM trips
        WHERE starttimestamp(trip) >= (SELECT t1 FROM t1)
          AND starttimestamp(trip) < (SELECT t FROM t)
          AND endtimestamp(trip) >= (SELECT t FROM t)
          AND endtimestamp(trip) <= (SELECT t2 FROM t2)
    )
SELECT first_half.first_half_count,
       second_half.second_half_count,
       crossed_t.crossed_t_count
FROM first_half, second_half, crossed_t;
```

Query 05

📋 We want to calculate the usage of edges by trips, i.e., to compute how many trips traversed each of the edges on the network, and create a table called “HeatMap” with the result. Write an SQL query to create and populate such table.

Solution

We create two spatial-temporal indexes to improve the performance of the query.

```
CREATE INDEX idx_roadsegments_segmentgeo ON roadsegments USING gist (segmentgeo);
CREATE INDEX idx_trips_trajectory ON trips USING gist (trajectory);
```

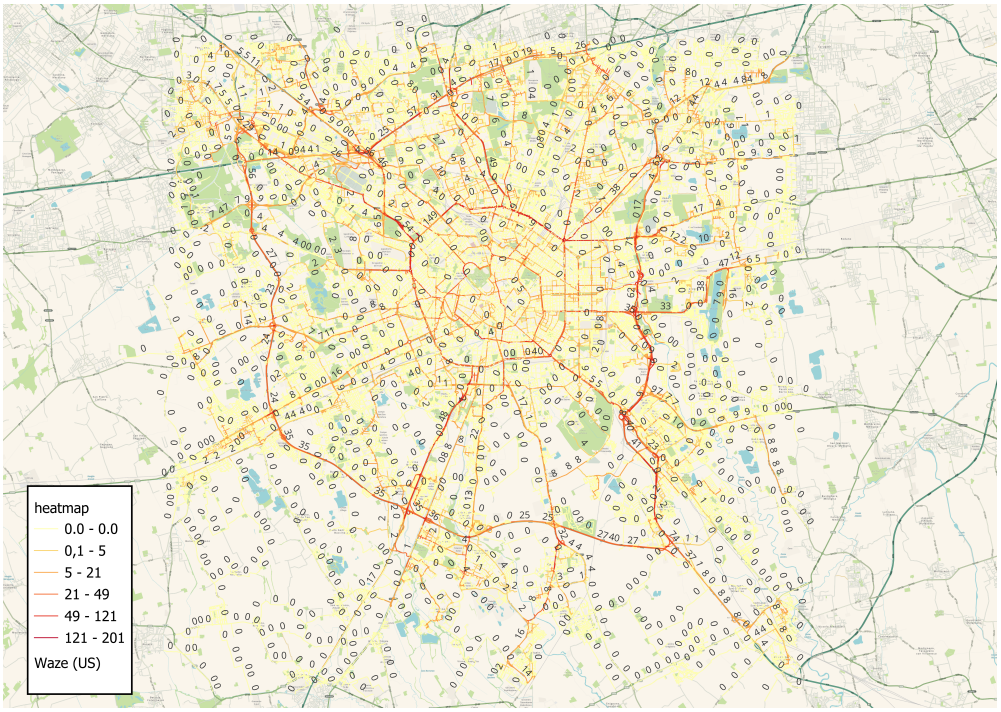
```
CREATE TABLE HeatMap AS (
    SELECT rs.segmentid as segmentid, rs.segmentgeo as segmentgeo,
        coalesce(count(t.tripid), 0) AS heat
    FROM roadsegments rs LEFT JOIN trips t ON st_intersects(t.trajectory, rs.segmentgeo)
    GROUP BY rs.segmentid, rs.segmentgeo
);
```

Query 06

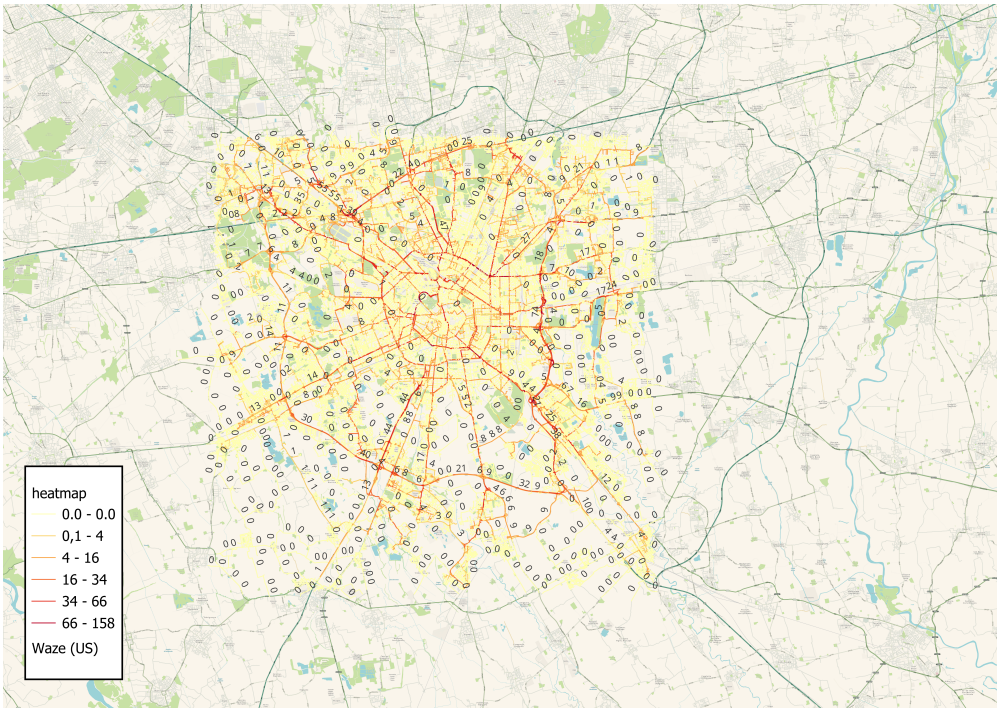
📋 Visualize in QGIS the information in table HeatMap. Decorate the visualization according to the usage of trips. Display the edges’s usage count in the map.

QGIS Output

Eco-friendly variant



Car-priority variant



Comparing both outputs, we can see that in the **car-priority variant** the cars drive more through the historic center of Milano, while the **eco-friendly variant** prioritize the outbounds streets, around the center.

Nevertheless, some streets are more frequented in the **eco-friendly variant** than in the **car-priority variant**, reaching a maximum of 201 and 158 hits each.

Query 07

📋 We want to categorize the type of trip in table “trips” adding four labels: work_home, home_work, leisure_weekday and leisure_weekend. You can use the table “Leisuretrips” for this.

a. Write an SQL query that computes the total number of trips of each type, according to the previous classification. Show also the min, max and average duration of these labelled trips.

b. Display in QGIS the different kinds of trips.

Solution

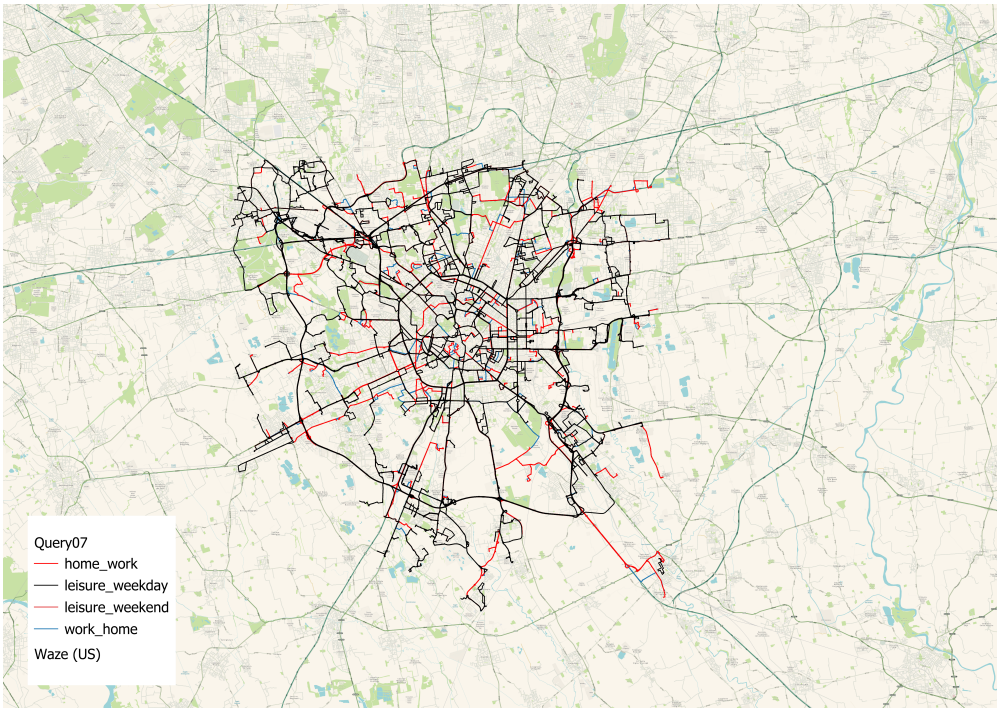
We create two spatial-temporal indexes to improve the performance of the query.

```
CREATE INDEX idx_nodes_geom ON nodes USING gist (geom);
CREATE INDEX idx_trips_trip ON trips USING gist (trip);
```

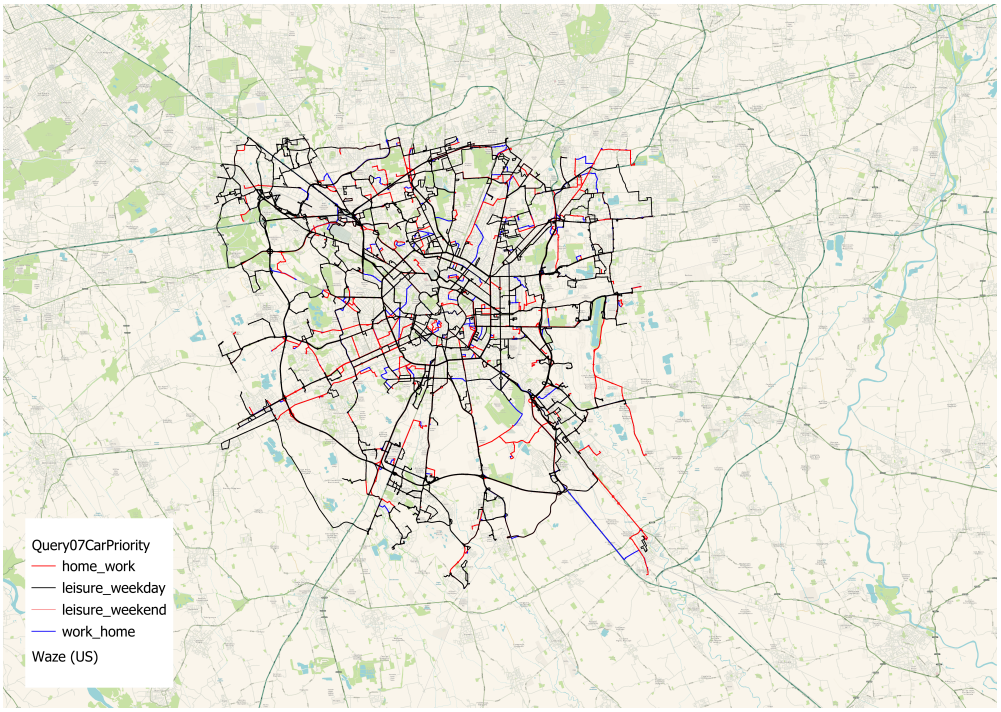
```
WITH trips_with_nodes AS (
   SELECT t.tripid, t.vehicleid, t.trajectory, duration(t.trip) as duration,
      t.startdate, n1.id AS start_id, n2.id AS end_id
   FROM trips t JOIN nodes n1 ON startvalue(t.trip) = n1.geom
                        JOIN nodes n2 ON endvalue(t.trip) = n2.geom
)
SELECT 'work_home' AS trip_type, COUNT(*) AS count, MIN(duration) AS min,
   MAX(duration) AS max, AVG(duration) AS avg, st_union(t.trajectory) as geom
FROM trips_with_nodes t JOIN vehiclenodes vn ON t.start_id = vn.worknode
   AND vn.vehicleid = t.vehicleid AND t.end_id = vn.homenode
UNION ALL
SELECT 'home_work' AS trip_type, COUNT(*) AS count, MIN(duration) AS min,
   MAX(duration) AS max, AVG(duration) AS avg, st_union(t.trajectory) as geom
FROM trips_with_nodes t JOIN vehiclenodes vn ON t.start_id = vn.homenode
   AND vn.vehicleid = t.vehicleid AND t.end_id = vn.worknode
UNION ALL
SELECT 'leisure_weekday' AS trip_type, COUNT(*) AS count, MIN(duration) AS min,
   MAX(duration) AS max, AVG(duration) AS avg, st_union(t.trajectory) as geom
FROM trips_with_nodes t JOIN leisuretrips lt ON t.start_id = lt.sourcenode
   AND t.end_id = lt.targetnode AND lt.vehicleid = t.vehicleid
WHERE EXTRACT(ISODOW FROM lt.startdate) <= 5
UNION ALL
SELECT 'leisure_weekend' AS trip_type, COUNT(*) AS count, MIN(duration) AS min,
   MAX(duration) AS max, AVG(duration) AS avg, st_union(t.trajectory) as geom
FROM trips_with_nodes t JOIN leisuretrips lt ON t.start_id = lt.sourcenode
   AND t.end_id = lt.targetnode AND lt.vehicleid = t.vehicleid
WHERE EXTRACT(ISODOW FROM lt.startdate) > 5;
```

QGIS Output

Eco-friendly variant



Car-priority variant



We can appreciate that in the **car-priority variant** the *work_home* trips differ more from the *home_work* trips than in the **eco-friendly variant**, as we can see more blue lines.

Query 08

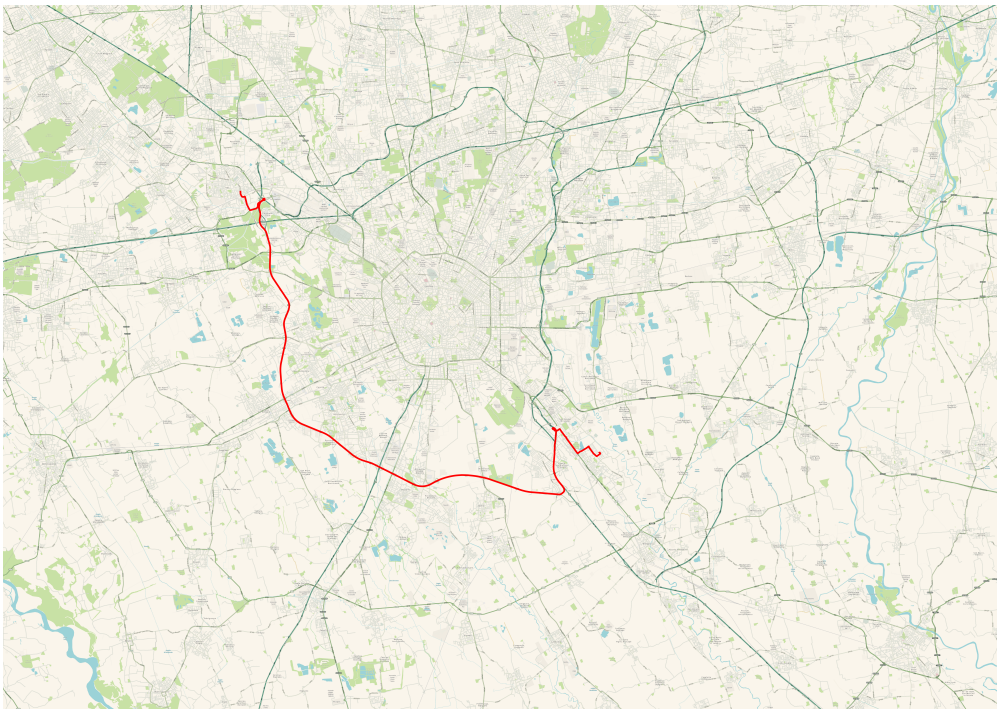
📋 Compute and visualize in QGIS the longest trip.

Solution

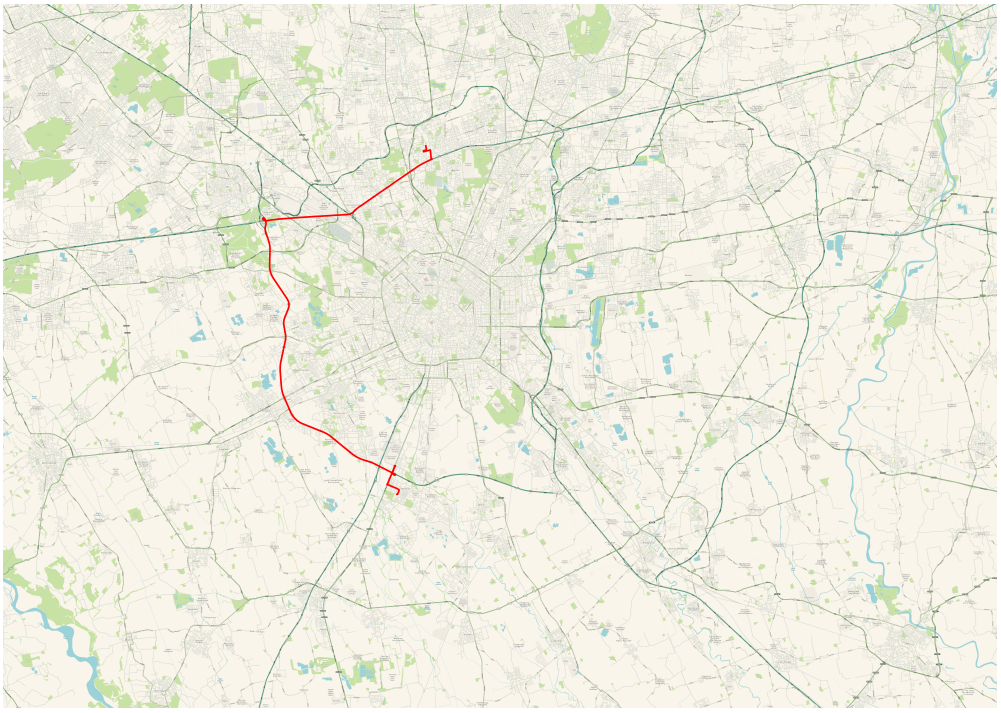
```
SELECT t.tripid, t.trajectory, st_length(t.trajectory) AS length
FROM trips t
ORDER BY st_length(t.trajectory) DESC
LIMIT 1;
```

QGIS Output

Eco-friendly variant



Car-priority variant



Query 09

📋 Write an SQL query to compute the vehicles that have crossed, in any trip, a region in table Regions during a period in table Periods. List three identifiers: VehicleId, periodID and RegionId. (Consider using the && operator to speed up the queries).

Solution

We create two spatial-temporal indexes to improve the performance of the query.

```
CREATE INDEX idx_regions_geom ON regions USING gist (geom);
CREATE INDEX idx_periods_period ON periods USING gist (period);
```

```
SELECT t.vehicleid, p.periodid, r.regionid
FROM trips t, regions r, periods p
WHERE t.trip && stbox(r.geom, p.period) AND
    eintersects(attime(t.trip, p.period), r.geom);
```

Query 10

📋 List the first time at which a vehicle visited a point in table Points

Solution

We create a spatial-temporal index to improve the performance of the query.

```
CREATE INDEX idx_points_geom ON points USING gist (geom);
```

```
SELECT t.vehicleid, p.pointid,
   min(starttimestamp(atgeometry(t.trip, p.geom))) as instant
FROM trips t, points p
WHERE atgeometry(t.trip, p.geom) IS NOT NULL
GROUP BY t.vehicleid, p.pointid;
```

Query 11

📋 Write an SQL query that computes the number of vehicles that were active at each period in table Periods

Solution

```
SELECT p.periodid, tcount(attime(t.trip, p.period)) as wcount
FROM trips t JOIN periods p ON t.trip && p.period
GROUP BY p.periodid
ORDER BY p.periodid;
```

Query 12

📋 Write an SQL query that computes the number of trips that were active during each hour in Jun 03, 2020

Solution

```
WITH jun_3_2020_hours AS (
   SELECT generate_series(timestamptz '2020-06-03 00:00:00',
                          timestamptz '2020-06-03 23:00:00', interval '1 hour') as hour
),
   jun_3_2020_hour_periods AS (
      SELECT span(hour, hour + interval '1 hour') AS period
      FROM jun_3_2020_hours
   )
SELECT p.period, coalesce(count(t.tripid), 0) AS trip_count
FROM jun_3_2020_hour_periods p LEFT JOIN trips t ON t.trip && p.period
GROUP BY p.period
ORDER BY p.period;
```

Query 13

📋 Write an SQL query that computes the overall traveled distances for each vehicle per period, during the periods in table Periods, in descending order

Solution

```
SELECT t.vehicleid, p.periodid, p.period,
    sum(st_length(trajectory(attime(t.trip, p.period)))) AS distance
FROM trips t JOIN periods p ON t.trip && p.period
GROUP BY t.vehicleid, p.periodid, p.period
ORDER BY distance DESC;
```