time_between_events

```
-- Compute the gaps
WITH request_gaps AS (
     SELECT date_created,
          -- lead or lag
          lag(date_created) OVER (ORDER BY date_created) AS previous,
          -- compute gap as date_created minus lead or lag
          date_created - lag(date_created) OVER (ORDER BY date_created) AS gap
      FROM evanston311)
-- Select the row with the maximum gap
SELECT *
 FROM request_gaps
-- Subquery to select maximum gap from request_gaps
 WHERE gap = (SELECT MAX(gap)
           FROM request_gaps);
```

Explanation:

- This SQL query finds the largest time gap between consecutive date_created entries in the evanston311 table. It uses a common table expression (CTE) called request_gaps to calculate the difference (gap) between each date_created and the one before it using the LAG() window function. Finally, it selects the row from request_gaps where this gap is the maximum.

```
-- Truncate the time to complete requests to the day
SELECT date_trunc('day', date_completed - date_created) AS completion_time,
-- Count requests with each truncated time
     COUNT(*)
 FROM evanston311
-- Where category is rats
 WHERE category = 'Rodents- Rats'
-- Group and order by the variable of interest
 GROUP BY completion_time
 ORDER BY COUNT;
```

Explanation:

- This SQL query analyzes data from the evanston311 table to determine the distribution of time taken to complete requests related to rats. It calculates the difference between date_completed and date_created, truncates this difference to the day, and then counts the number of requests falling into each day-based completion time category. The results are ordered by the count of requests.

```
SELECT category,
     -- Compute average completion time per category
     AVG(date_completed - date_created) AS avg_completion_time
 FROM evanston311
-- Where completion time is less than the 95th percentile value
 WHERE date_completed - date_created <
-- Compute the 95th percentile of completion time in a subquery
       (SELECT percentile_disc(0.95) WITHIN GROUP (ORDER BY date_completed - date_created)
         FROM evanston311)
 GROUP BY category
-- Order the results
 ORDER BY avg_completion_time DESC;
```

Explanation:

- This SQL query calculates the average completion time for each category in the evanston311 table, but only considers completion times below the 95th percentile. It does this by first calculating the 95th percentile of all completion times using a subquery, and then filtering the main query to include only entries with completion times below that threshold. Finally, it groups the results by category and orders them by average completion time in descending order.

```
-- Compute correlation (corr) between
-- avg_completion time and count from the subquery
SELECT corr(avg_completion, count)
 -- Convert date_created to its month with date_trunc
 FROM (SELECT date_trunc('month', date_created) AS month,
           -- Compute average completion time in number of seconds
           AVG(EXTRACT(epoch FROM date_completed - date_created)) AS avg_completion,
           -- Count requests per month
           count(*) AS count
       FROM evanston311
      -- Limit to rodents
      WHERE category='Rodents- Rats'
      -- Group by month, created above
      GROUP BY month)
      -- Required alias for subquery
      AS monthly_avgs;
```

Explanation:

- This SQL query calculates the correlation between the average completion time and the number of requests for rodent-related issues in the evanston311 table, grouped by month. It first creates a subquery to calculate the monthly average completion time (in seconds) and the request count. Then, the outer query uses the corr() function to compute the correlation coefficient between these two aggregated values.

```
-- Compute monthly counts of requests created
WITH created AS (
    SELECT date_trunc('month', date_created) AS month,
        count(*) AS created_count
     FROM evanston311
    WHERE category='Rodents- Rats'
    GROUP BY month),
-- Compute monthly counts of requests completed
   completed AS (
    SELECT date_trunc('month', date_completed) AS month,
        count(*) AS completed_count
     FROM evanston311
    WHERE category='Rodents- Rats'
    GROUP BY month)
-- Join monthly created and completed counts
SELECT created.month,
    created_count,
    completed_count
 FROM created
    INNER JOIN completed
    ON created.month=completed.month
 ORDER BY created.month;
```

Explanation:

- This SQL query calculates and compares the monthly counts of created and completed requests for “Rodents-Rats” category from the evanston311 table. It uses Common Table Expressions (CTEs) called created and completed to separately aggregate the counts by month. Finally, it joins these CTEs to display the created and completed counts side-by-side for each month.