aggregate_date_time_series

```
SELECT day
-- 1) Subquery to generate all dates
-- from min to max date_created
 FROM (SELECT generate_series(MIN(date_created),
                    MAX(date_created),
                    '1 day')::date AS day
      -- What table is date_created in?
      FROM evanston311) AS all_dates
-- 4) Select dates (day from above) that are NOT IN the subquery
 WHERE day NOT IN
    -- 2) Subquery to select all date_created values as dates
    (SELECT date_created::date
      FROM evanston311);
```

Explanation:

- This SQL query identifies dates within a range (defined by the minimum and maximum date_created values in the evanston311 table) that are *not* present in the date_created column of the same table. It does this by generating a series of all dates within the range and then filtering out those dates that exist in the table's date_created column. The result shows any gaps in the date_created data.

```
-- Generate 6 month bins covering 2016-01-01 to 2018-06-30

-- Create lower bounds of bins
SELECT generate_series('2016-01-01',  -- First bin lower value
              '2018-06-30',  -- Last bin lower value
              '6 months'::interval) AS lower,
-- Create upper bounds of bins
    generate_series('2016-07-01',  -- First bin upper value
              '2018-12-30',  -- Last bin upper value
              '6 months'::interval) AS upper;
```

Explanation:

- This SQL query uses the generate_series function to create a series of date ranges (bins) with a 6-month interval. It generates two series: one for the lower bounds of each bin and another for the upper bounds. The result is a table showing the start and end dates of 6-month periods from 2016-01-01 to 2018-06-30.

```
-- Count number of requests made per day
SELECT day, COUNT(date_created) AS count
```

```
-- Use a daily series from 2016-01-01 to 2018-06-30
-- to include days with no requests
 FROM (SELECT generate_series('2016-01-01',  -- series start date
                 '2018-06-30',  -- series end date
                 '1 day'::interval)::date AS day) AS daily_series
    LEFT JOIN evanston311
    -- match day from above (which is a date) to date_created
    ON day = date_created::date
 GROUP BY day;
```

Explanation:

- This SQL query counts the number of requests made each day within a specified date range. It generates a series of dates and then performs a LEFT JOIN with the evanston311 table (presumably containing request data) to count requests for each day, including days with zero requests. The generate_series function creates the date range, and the LEFT JOIN ensures that all days are included in the result.

```
-- Bins from Step 1
WITH bins AS (
   SELECT generate_series('2016-01-01',
                '2018-01-01',
                '6 months'::interval) AS lower,
        generate_series('2016-07-01',
                '2018-07-01',
                '6 months'::interval) AS upper),
-- Daily counts from Step 2
   daily_counts AS (
   SELECT day, count(date_created) AS count
    FROM (SELECT generate_series('2016-01-01',
                    '2018-06-30',
                    '1 day'::interval)::date AS day) AS daily_series
        LEFT JOIN evanston311
        ON day = date_created::date
    GROUP BY day)
-- Select bin bounds
SELECT lower,
    upper,
    -- Compute median of count for each bin
    percentile_disc(0.5) WITHIN GROUP (ORDER BY COUNT) AS median
 -- Join bins and daily_counts
 FROM bins
    LEFT JOIN daily_counts
    -- Where the day is between the bin bounds
    ON day >= lower
      AND day < upper
 -- Group by bin bounds
```

```sql
GROUP BY lower, upper
ORDER BY lower;
```

Explanation:

- This SQL query calculates the median daily count of events from the evanston311 table for 6-month periods between 2016-01-01 and 2018-01-01. It first creates a series of 6-month bins (bins CTE). Then, it calculates the daily counts (daily_counts CTE). Finally, it joins these two CTEs to find the median count within each 6-month bin using the percentile_disc function. The result shows the start and end date of each bin and its corresponding median daily count.

```sql
-- generate series with all days from 2016-01-01 to 2018-06-30
WITH all_days AS (
  SELECT generate_series('2016-01-01', '2018-06-30', '1 day'::interval) AS date
),
-- Subquery to compute daily counts
daily_count AS (
  SELECT date_trunc('day', date_created) AS day,
      count(*) AS count
  FROM evanston311
  GROUP BY day
)
-- Aggregate daily counts by month using date_trunc
SELECT date_trunc('month', date) AS month,
    -- Use coalesce to replace NULL count values with 0
    avg(coalesce(count, 0)) AS average
FROM all_days
   LEFT JOIN daily_count
   -- Joining condition
   ON all_days.date = daily_count.day
GROUP BY month
ORDER BY month;
```

Explanation:

- This SQL query calculates the average daily count of records from the evanston311 table for each month between 2016-01-01 and 2018-06-30. It first generates a series of all dates within this range. Then, it counts the records in evanston311 for each day. Finally, it joins these daily counts with the date series, calculates the average count per month using AVG and COALESCE (to handle months with zero counts), and orders the results by month.