exploring_distributions

```sql
-- Truncate employees
SELECT trunc(employees, -5) AS employee_bin,
    -- Count number of companies with each truncated value
    COUNT(*)
 FROM fortune500
-- Use alias to group
 GROUP BY employee_bin
-- Use alias to order
 ORDER BY employee_bin;
```

Explanation:

- This SQL query analyzes the fortune500 table, which presumably contains information about Fortune 500 companies, including the number of employees (employees column). It groups companies based on the number of employees, truncated to the nearest 100,000 (using trunc(employees, -5)). The query then counts the number of companies in each group and displays the truncated employee count alongside the company count. The results are ordered by the truncated employee count.

```sql
-- Truncate employees
SELECT
  trunc(employees, -4) AS employee_bin,  -- Truncate the 'employees' column to the nearest 10000, creating bins
  COUNT(*)                -- Count the number of companies in each bin
FROM
  fortune500
WHERE
  employees < '100000'          -- Filter for companies with less than 100000 employees
GROUP BY
  employee_bin              -- Group the results by the employee bins
ORDER BY
  employee_bin;             -- Order the results by the employee bins
```

Explanation:

- This SQL query analyzes the fortune500 table to determine the distribution of companies based on their employee count. It groups companies into bins of 10,000 employees using the TRUNC function, counts the number of companies in each bin, and then presents the results ordered by the employee bins. The WHERE clause filters the data to include only companies with fewer than 100,000 employees.

```sql
-- Select the min and max of question_count
SELECT MIN(question_count),
    MAX(question_count)
 -- From what table?
 FROM stackoverflow
-- For tag dropbox
WHERE tag='dropbox';
```

Explanation:

- This SQL query selects the minimum and maximum values of the question_count column from the stackoverflow table, but only for rows where the tag column is equal to 'dropbox'. It efficiently finds the range of question counts associated with the 'dropbox' tag.

```sql
-- Create lower and upper bounds of bins
SELECT generate_series(2200, 3050, 50) AS lower,
    generate_series(2250, 3100, 50) AS upper;
```

Explanation:

- This SQL query uses the generate_series function to create a series of numbers representing the lower and upper bounds of bins. Each bin has a width of 50. The lower bounds range from 2200 to 3050, while the upper bounds range from 2250 to 3100, ensuring each bin is 50 units wide. The result is a table with two columns, lower and upper, showing the boundaries of each bin.

```sql
-- Bins created in Step 2
WITH bins AS (
    SELECT generate_series(2200, 3050, 50) AS lower,
        generate_series(2250, 3100, 50) AS upper),
   -- Subset stackoverflow to just tag dropbox (Step 1)
   dropbox AS (
    SELECT question_count
     FROM stackoverflow
    WHERE tag='dropbox')
-- Select columns for result
-- What column are you counting to summarize?
SELECT lower, upper, count(question_count)
 FROM bins  -- Created above
    -- Join to dropbox (created above), keeping all rows from the bins table in the join
    LEFT JOIN dropbox
    -- Compare question_count to lower and upper
     ON dropbox.question_count >= bins.lower
    AND dropbox.question_count < bins.upper
-- Group by lower and upper to count values in each bin
GROUP BY lower, upper
```

```
-- Order by lower to put bins in order
ORDER BY lower;
```

Explanation:

- This SQL query performs a histogram analysis on the question_count from a stackoverflow table, specifically for questions tagged ‘dropbox’. It creates bins (ranges of values) using generate_series and then counts the number of question_count values falling into each bin using a LEFT JOIN to ensure all bins are included in the result, even if empty. The final result shows the lower and upper bounds of each bin and the count of question_count values within that bin.