Milestone 10 | Construction Job Demand

INTRODUCTION: This is a Milestone for students looking to level up their SQL skills with subqueries! Subqueries allow us to write efficient queries and to perform more meaningful analysis, highlighting to employers a deeper understanding of the nuances of SQL.

In this Milestone, you'll be using subqueries to analyze data regarding construction jobs and the role weather might play in driving future demand. Companies in this space need to be able to assess what future demand will look like in order to allocate resources and teams accordingly. Sending a small team to a location may mean potential customers will go with a competitor because they can get the job done in a shorter time frame due to your limited resources.

HOW IT WORKS: Follow the prompts in the questions below to investigate your data. Post your answers in the provided boxes: the **yellow boxes** for the queries you write, and **blue boxes** for text-based answers. When you're done, export your document as a pdf file and submit it on the Milestone page – see instructions for creating a PDF at the end of the Milestone. **But understand that you will not be submitting this Milestone.**

RESOURCES: If you need hints on the Milestone or are feeling stuck, there are multiple ways of getting help. Attend Drop-In Hours to work on these problems with your peers, or reach out to the HelpHub if you have questions. Good luck!

PROMPT: Congratulations, you've landed an internship on the Data Analysis Team at Hover. Your manager is interested in the demand of jobs. She has a hunch that severe weather events have an effect on the number of job requests, but she'd like you to help her prove her hunch is true with data.

SQL App: Here's that link to our specialized SQL app, where you'll write your SQL queries and interact with the data.

— Data Set **Description**

The data needed for this Milestone (`hover.*`) comes from two different sources.

The first data source (`jobs`) consists of historical data from a software solution provider for construction companies:

- **`job_identifier`** - The unique ID for a job
- **`organization_id`** - ID for the account that purchased the job
- **`job_location_city`** - The city where the job was located
- **`job_location_region_code`** - The state where the job was located
- **`job_first_upload_complete_datetime`** - The date on which the customer uploaded photos for the job.
- **`job_deliverable`** - the type of job that the customer requested. Either complete (full building) or roof.

The second dataset (`weather`) comes from the NOAA Storm Prediction Center and catalogs adverse weather events across the United States:

- **`comments`** - A description of the weather event
- **`county`** - The county of the weather event
- **`state`** - The state of the weather event
- **`location`** - The address of the weather event
- **`longitude`** - The longitude of the weather event
- **`latitude`** - The latitude of the weather event
- **`datetime`** - The datetime of the weather event
- **`composite_key`** - Unique key for the weather event, combining the timestamp, longitude, and latitude of the event

---

— **Task 1:** Explore the jobs data.

Your manager would like to find out more information regarding when customers make requests.

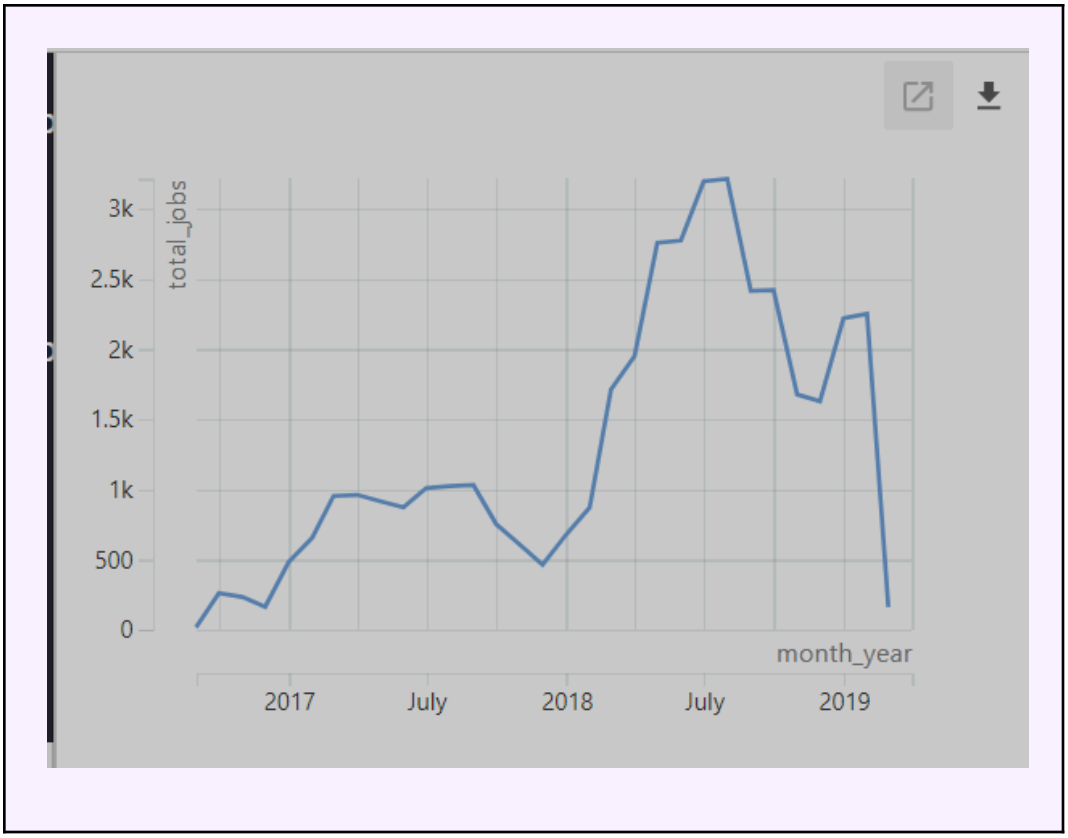
A. Write a query that returns the total number of jobs at the monthly level for each year (i.e. Sept 2016, Oct 2016, etc.). You'll need to use the `date_trunc`

function in SQL with the **`job_first_upload_complete_datetime`** variable. When done correctly, your output will have 31 rows.

(paste your query below 👇)

```
SELECT
  DATE_TRUNC('month', job_first_upload_complete_datetime) AS
month_year,
  COUNT(job_identifier) AS total_jobs
FROM
  hover.jobs
GROUP BY
  DATE_TRUNC('month', job_first_upload_complete_datetime)
ORDER BY
  month_year;
```

B. Use the SQL app's built-in visualizer to graph the task in part A. Is there any seasonality to the job requests? Seasonality can be described as a pattern that repeats itself over cycles of time, such as yearly. Is there a season that has more requests than others?



(write your **answer** below 👇)

Regarding seasonality there is one constant where December into the new year requests increase. The season with more requests than all the others is the summer months of 2018.

C. Level Up[2] - One of the reasons why we didn't look at data aggregated by month (without the year) is because the first month of the data is September 2016 and the last month of data is March 2019, so every month wouldn't have appeared an equal number of times in a `GROUP BY`. Can you write a query that counts how many times each month has appeared in the data? Hint: You'll need both the date_part and the date_trunc function here. You should `GROUP BY` month, using the date_part function. If done correctly, each month should have a count of 2 or 3.

(paste your query below 👇)

```
SELECT
  DATE_PART('month', job_first_upload_complete_datetime) AS
month,
  COUNT(
    DISTINCT DATE_TRUNC('month',
job_first_upload_complete_datetime)
  ) AS month_count
FROM
  hover.jobs
GROUP BY
  DATE_PART('month', job_first_upload_complete_datetime)
ORDER BY
  month;
```

– **Task 2:** Explore the weather data.

In Task 1, you discovered that there are seasonal patterns to when customers submit job requests. Now you'll investigate the weather data and see how that can help you further with your analysis.

A. The entire weather dataset consists of "adverse weather events", e.g., tornados, fallen trees, sustained high gusts of wind, etc. Write a query that counts the number of adverse weather events for each month and year of the data (i.e. Sept 2016, Oct 2016, etc.).

(paste your query below 👇)

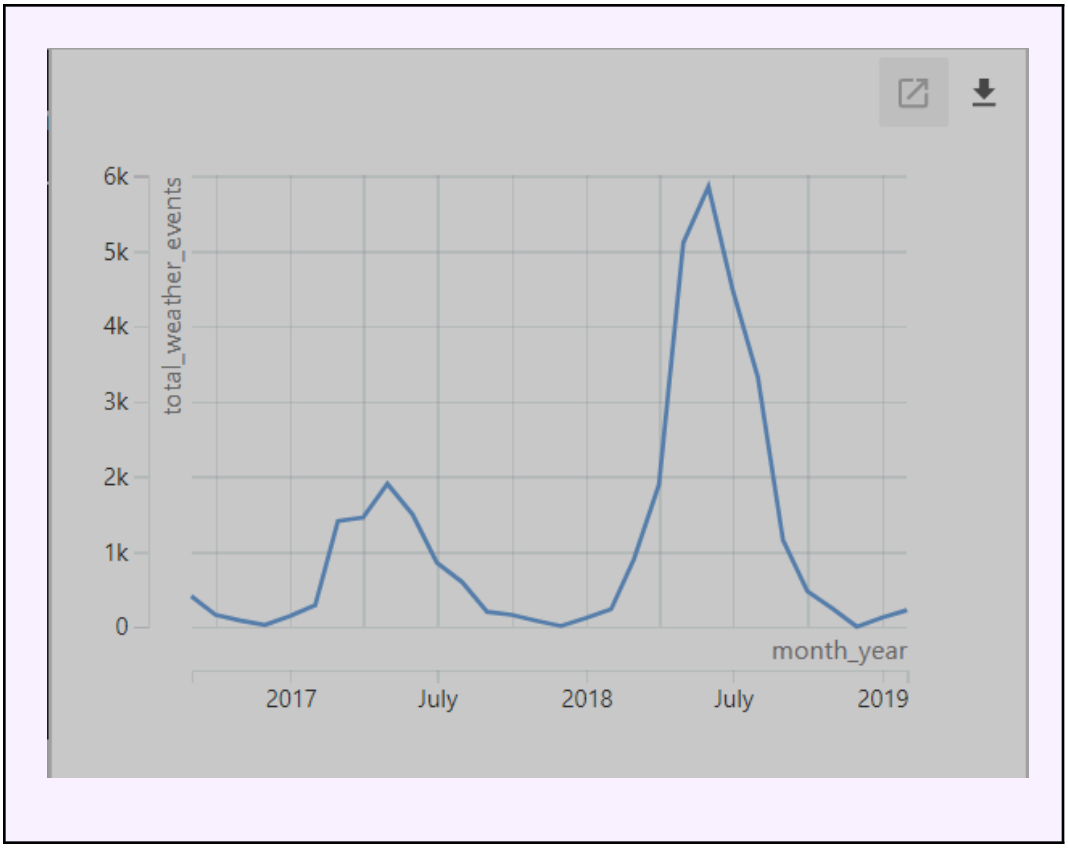
```
SELECT
  DATE_TRUNC('month', datetime) AS month_year,
  COUNT(composite_key) AS total_weather_events
FROM
  hover.weather
GROUP BY
```

```
  DATE_TRUNC('month', datetime)
ORDER BY
  month_year;
```

B. Modify your query in 2A to filter out any information prior to September 2016 (the start of the job request data). Visualize the data using the built-in visualizer.

(paste your query below 👇)

```
SELECT
  DATE_TRUNC('month', datetime) AS month_year,
  COUNT(composite_key) AS total_weather_events
FROM
  hover.weather
WHERE
  datetime >= '2016-09-01'
GROUP BY
  DATE_TRUNC('month', datetime)
ORDER BY
  month_year;
```

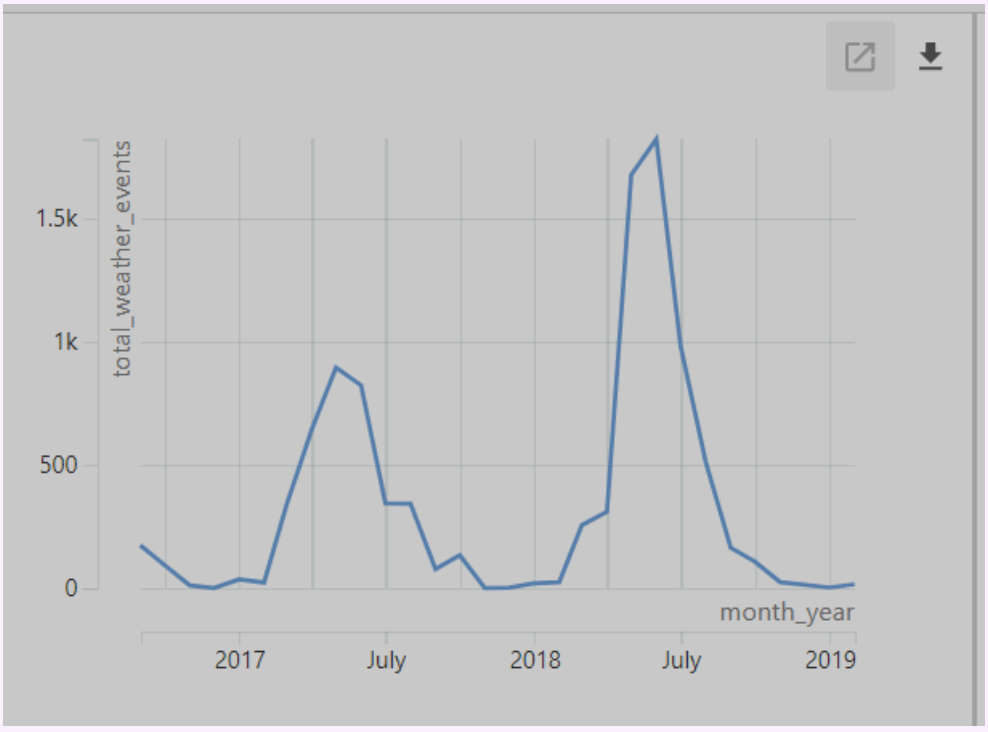


C. The weather data includes values for all 50 states. Modify your query once more so that it only shows information from the states that are seen in the jobs data. Visualize the filtered data again with the built-in visualizer and compare the filtered data to the original data from 2B. Did the filtering change anything about the pattern? Note: Your query should **NOT** use a `JOIN` clause, but instead use a **subquery**!.

(paste your query below 👇)

```
SELECT
  DATE_TRUNC('month', datetime) AS month_year,
  COUNT(composite_key) AS total_weather_events
FROM
  hover.weather
WHERE
  datetime >= '2016-09-01'
  AND state IN (
```

```
    SELECT
      DISTINCT job_location_region_code
    FROM
      hover.jobs
  )
GROUP BY
  DATE_TRUNC('month', datetime)
ORDER BY
  month_year;
```



D. Write a few-sentence summary describing the relationship between the job requests and weather events you observed in Tasks 1 and 2.

(write your **answer** below 👇)

The job events and weather events are closely aligned in the dataset. Especially during the earlier years in the dataset, it follows roughly the same path, but near the latter half, the weather falls completely before the requests do.

— **Task 3:** Does weather affect demand for jobs?

Given the perceived relationship between the weather event graph and job requests graph, you will now investigate if there is indeed a relationship between adverse weather events and job requests.

A colleague of yours already wrote a query that returns the total number of weather events grouped at the weekly level for each state in the weather data. You'll build off their work.

A. Write a query that performs a `JOIN` on the `jobs` table and the `weekly_weather_events` table your colleague created. Since you are only interested in the weeks that have had an adverse weather event, use an `INNER JOIN` to match the tables on both the week timestamp AND the state. If you don't do this, you will end up with output crossing states, e.g. requests in TX incorrectly joined to events in KS.

`SELECT` the following columns from the jobs table in your query:

- **`job_deliverable`,**
- **`job_location_region_code`,**
- **`job_first_upload_complete_datetime`**, truncated down to the 'week' level. Alias this column as **`job_ts`**.

And from the `weekly_weather_events` table, you will need the following:

- **`n_weather_events`**

Remember to use the function and NOT the alias `job_ts` when joining to the `weather_ts column`, otherwise you will get an error. If done correctly, your output table will have 25,257 rows.

(paste your query below 👇)

```
SELECT
  j.job_deliverable,
  j.job_location_region_code,
  DATE_TRUNC('week', j.job_first_upload_complete_datetime) AS
job_ts,
  w.n_weather_events
FROM
  hover.jobs j
  INNER JOIN hover.weekly_weather_events w ON
DATE_TRUNC('week', j.job_first_upload_complete_datetime) =
w.weather_ts
  AND j.job_location_region_code = w.state
```

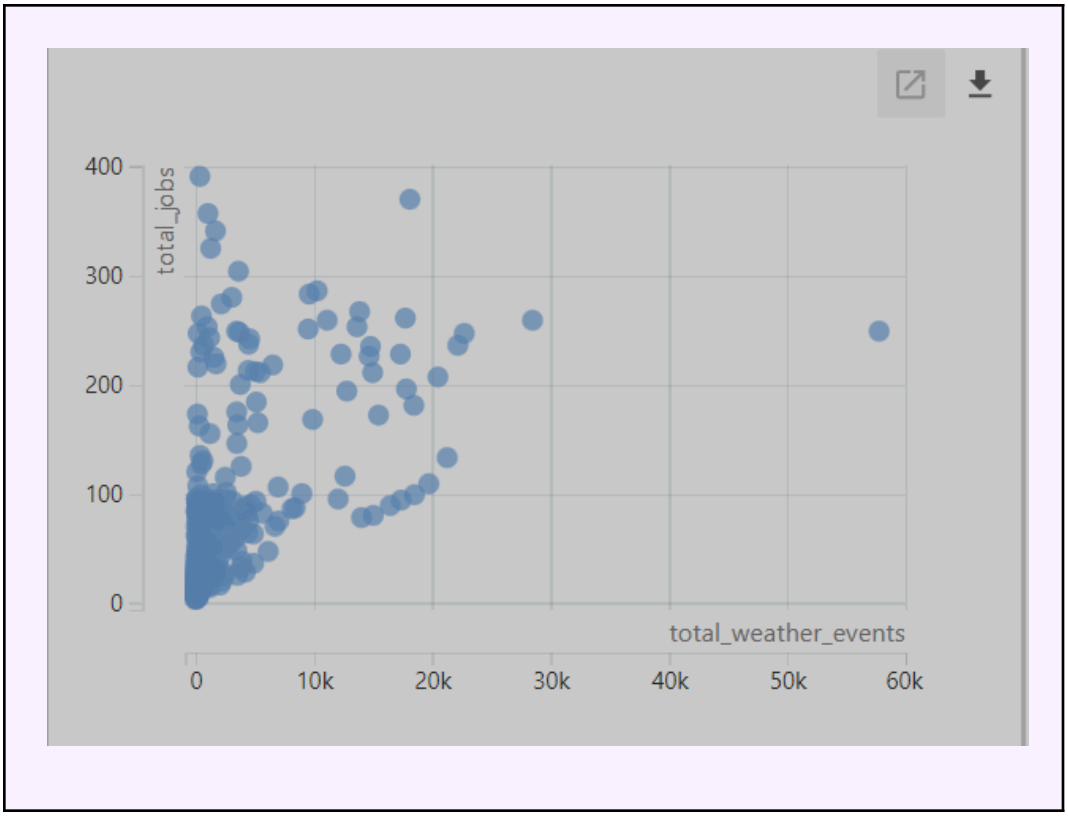
B. Use your query in 3A as a subquery in a new query that counts the total number of jobs and the total number of weather events for each state and week. The variable `job_ts` can be used to count the total number of jobs. Order your output alphabetically by state. If done correctly, your output table should have 320 rows.

(paste your query below 👇)

```
SELECT
  state,
  job_ts,
  COUNT(job_deliverable) AS total_jobs,
  SUM(n_weather_events) AS total_weather_events
FROM
  (
    SELECT
```

```
      j.job_deliverable,
      j.job_location_region_code AS state,
      DATE_TRUNC('week',
j.job_first_upload_complete_datetime) AS job_ts,
      w.n_weather_events
    FROM
      hover.jobs j
      INNER JOIN hover.weekly_weather_events w ON
DATE_TRUNC('week', j.job_first_upload_complete_datetime) =
w.weather_ts
      AND j.job_location_region_code = w.state
  ) AS subquery
GROUP BY
  state,
  job_ts
ORDER BY
  state;
```

C. Now you are ready to determine whether there is a relationship between adverse weather events and job requests. Using the built-in visualizer, create a scatterplot of your data. On your x-axis you should have the total number of weather events and on the y-axis you should have the total number of job requests. Check the box for “show trendline”. Is there a relationship between adverse weather events and job requests? If so, what kind of relationship is it?



(write your **answer** below 👇)

The data seems to be correlated or have a trend, as low weather events seem to correlate to lower total jobs. It is very loose but does show signs of a trend. There is a very notable outlier with Texas on the far right side with a huge amount of weather events. Overall, there is some indication that more weather events could lead to more job requests, but the spread of the data points suggests that other factors are also at play that go beyond just the data we have selected for the visualization.

— Submission

Great work completing this Milestone! To submit your completed Milestone, you will need to download / export this document as a PDF and then upload it to the Milestone submission page. You can find the option to download as a PDF from the File menu in the upper-left corner of the Google Doc interface.