```
       FROM sales
       GROUP BY date
       ORDER BY date;
+------------+-----------+
| date       | sum_price |
+------------+-----------+
| 2017-01-03 |    300.00 |
| 2017-01-06 |     50.00 |
| 2017-01-08 |    180.00 |
| 2017-01-10 |      5.00 |
+------------+-----------+
```

However, that result contains “holes” for dates not represented in the range of dates spanned by the table. A result that represents all dates in the range can be produced using a recursive CTE to generate that set of dates, joined with a `LEFT JOIN` to the sales data.

Here is the CTE to generate the date range series:

```
WITH RECURSIVE dates (date) AS
(
  SELECT MIN(date) FROM sales
  UNION ALL
  SELECT date + INTERVAL 1 DAY FROM dates
  WHERE date + INTERVAL 1 DAY <= (SELECT MAX(date) FROM sales)
)
SELECT * FROM dates;
```

The CTE produces this result:

```
+------------+
| date       |
+------------+
| 2017-01-03 |
| 2017-01-04 |
| 2017-01-05 |
| 2017-01-06 |
| 2017-01-07 |
| 2017-01-08 |
| 2017-01-09 |
| 2017-01-10 |
+------------+
```

How the CTE works:

- The nonrecursive `SELECT` produces the lowest date in the date range spanned by the `sales` table.
- Each row produced by the recursive `SELECT` adds one day to the date produced by the previous row.
- Recursion ends after the dates reach the highest date in the date range spanned by the `sales` table.

Joining the CTE with a `LEFT JOIN` against the `sales` table produces the sales summary with a row for each date in the range:

```
WITH RECURSIVE dates (date) AS
(
  SELECT MIN(date) FROM sales
  UNION ALL
  SELECT date + INTERVAL 1 DAY FROM dates
  WHERE date + INTERVAL 1 DAY <= (SELECT MAX(date) FROM sales)
)
SELECT dates.date, COALESCE(SUM(price), 0) AS sum_price
FROM dates LEFT JOIN sales ON dates.date = sales.date
GROUP BY dates.date
```