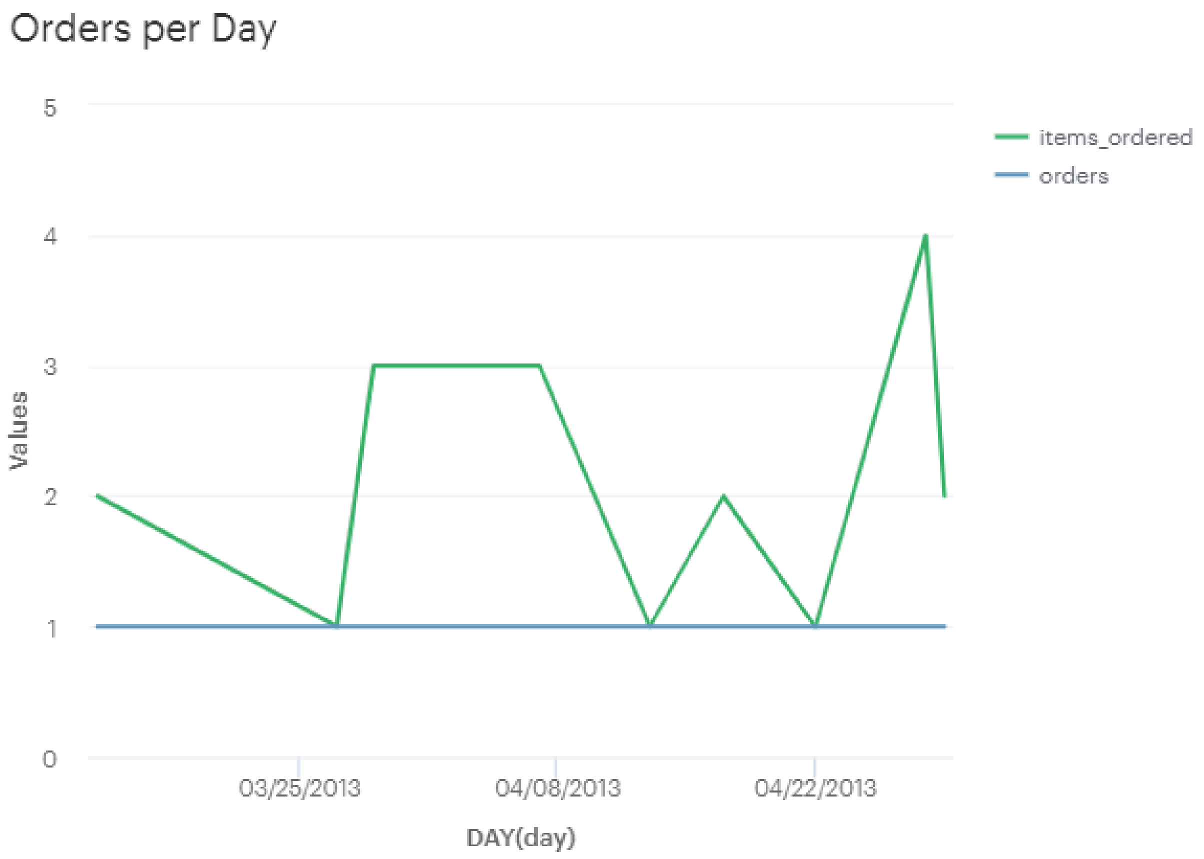
Create a Roll-up Table

Exercise 1: Create a sub-table of orders per day. Make sure you decide whether you are counting invoices or line items.

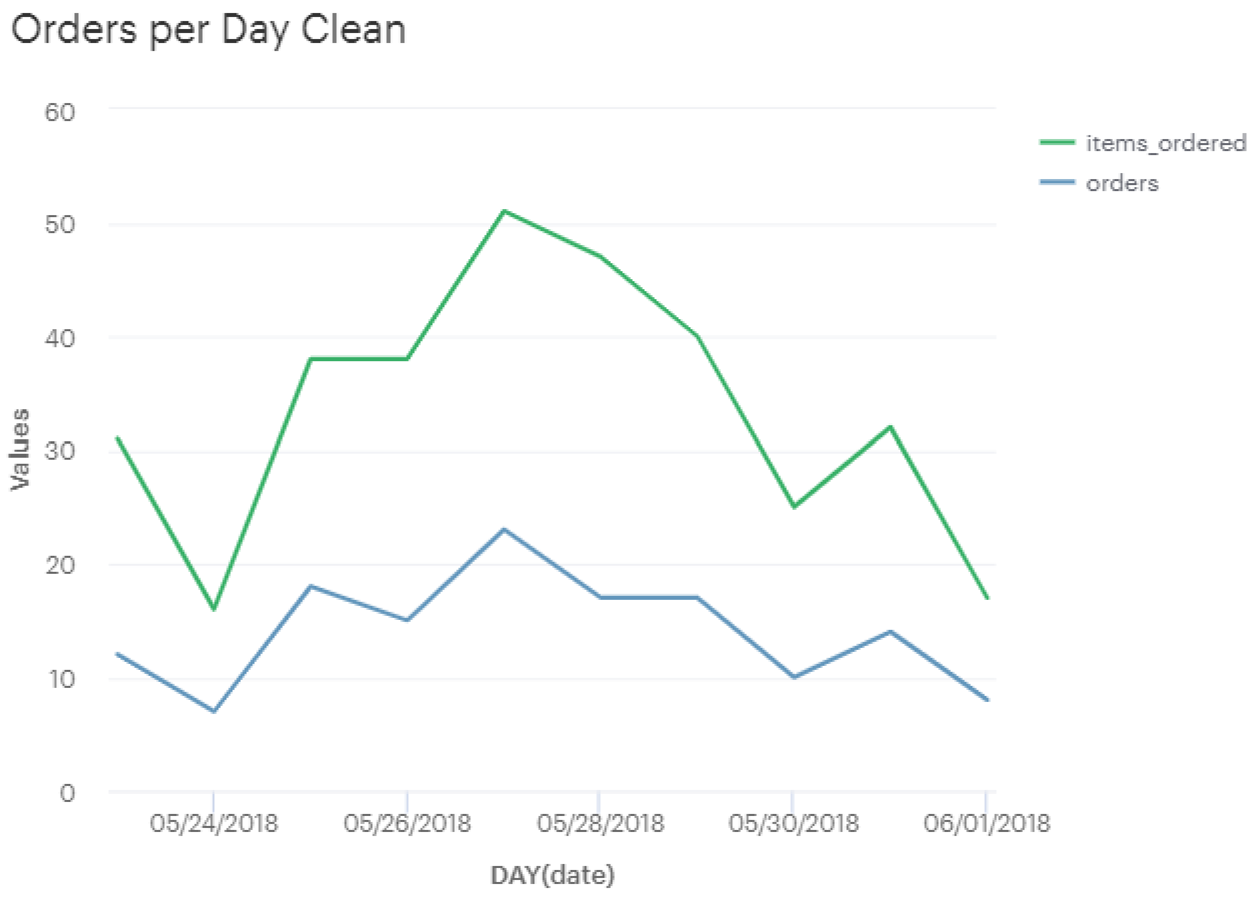
```
select
  date(orders.paid_at)          as day,
  count(distinct invoice_id)    as orders,
  count(distinct line_item_id)  as items_ordered
from
  dsv1069.orders
group by
  date(orders.paid_at)
```



Exercise 2: “Check your joins”. We are still trying to count orders per day. In this step join the sub table from the previous exercise to the dates rollup table so we can get a row for every date. Check that the join works by just running a “select *” query.

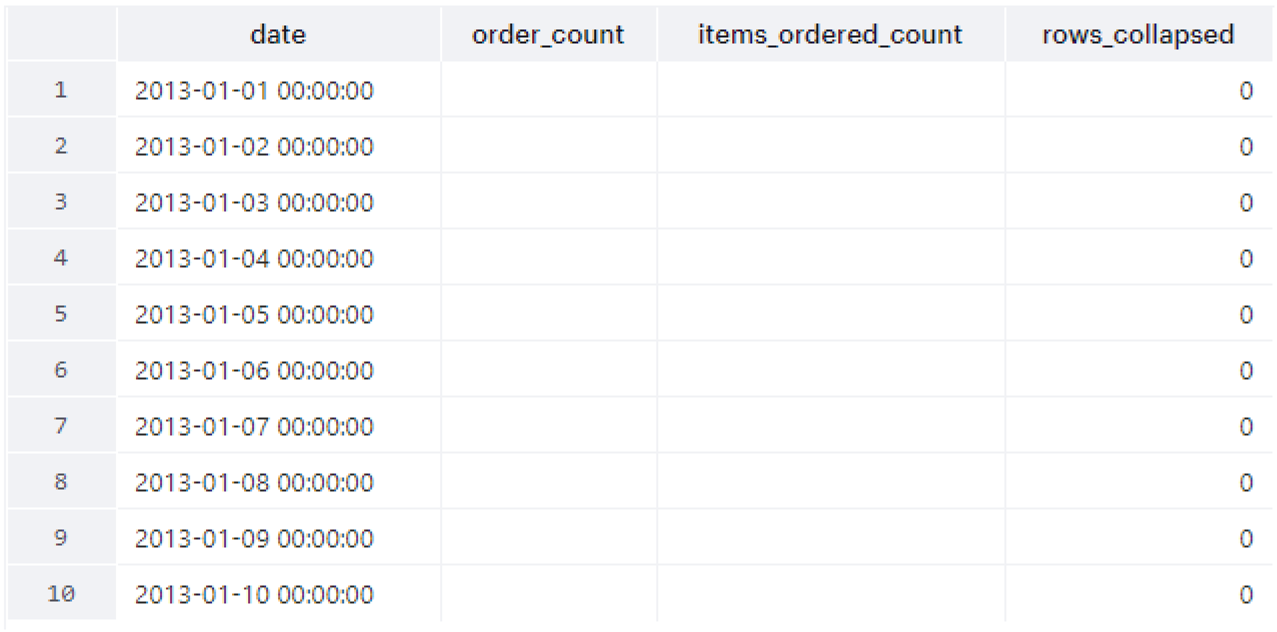
```
select
*
from
  dsv1069.dates_rollup
left outer join
  (
  select
    date(orders.paid_at)          as day,
    count(distinct invoice_id)    as orders,
    count(distinct line_item_id)  as items_ordered
  from
    dsv1069.orders
  group by
    date(orders.paid_at)
  ) daily_orders
on
```

```
  dates_rollup.date = daily_orders.day
```



Exercise 3: “Clean up your Columns” In this step be sure to specify the columns you actually want to return, and if necessary do any aggregation needed to get a count of the orders made per day.

```
select
  dates_rollup.date,
  sum(orders)         as order_count,
  sum(items_ordered)  as items_ordered_count,
  count(day)          as rows_collapsed
from
  dsv1069.dates_rollup
left outer join
  (
  select
    date(orders.paid_at)          as day,
    count(distinct invoice_id)    as orders,
    count(distinct line_item_id)  as items_ordered
  from
    dsv1069.orders
  group by
    date(orders.paid_at)
  ) daily_orders
on
  dates_rollup.date = daily_orders.day
group by
  dates_rollup.date
```

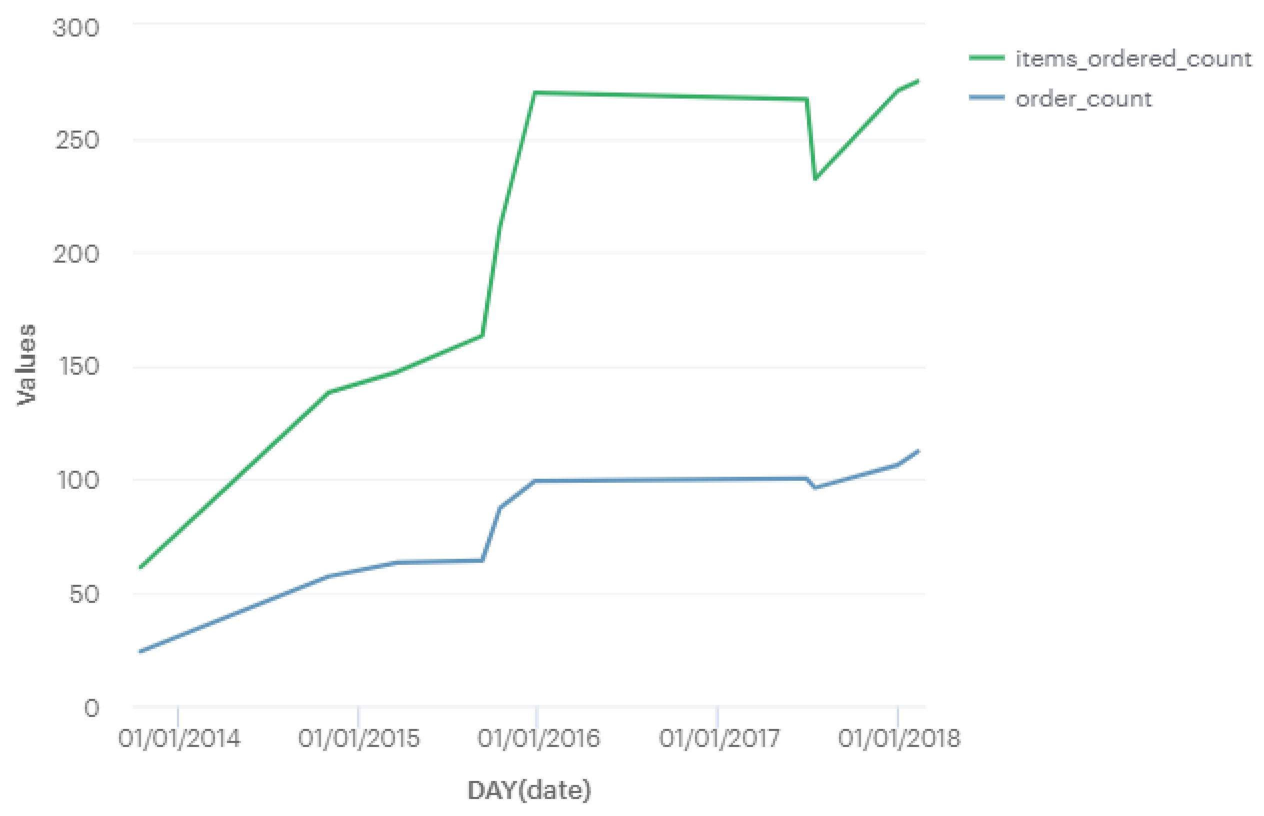


	date	order_count	items_ordered_count	rows_collapsed
1	2013-01-01 00:00:00			0
2	2013-01-02 00:00:00			0
3	2013-01-03 00:00:00			0
4	2013-01-04 00:00:00			0
5	2013-01-05 00:00:00			0
6	2013-01-06 00:00:00			0
7	2013-01-07 00:00:00			0
8	2013-01-08 00:00:00			0
9	2013-01-09 00:00:00			0
10	2013-01-10 00:00:00			0

Exercise 4: Weekly Rollup. Figure out which parts of the JOIN condition need to be edited create 7 day rolling orders table.

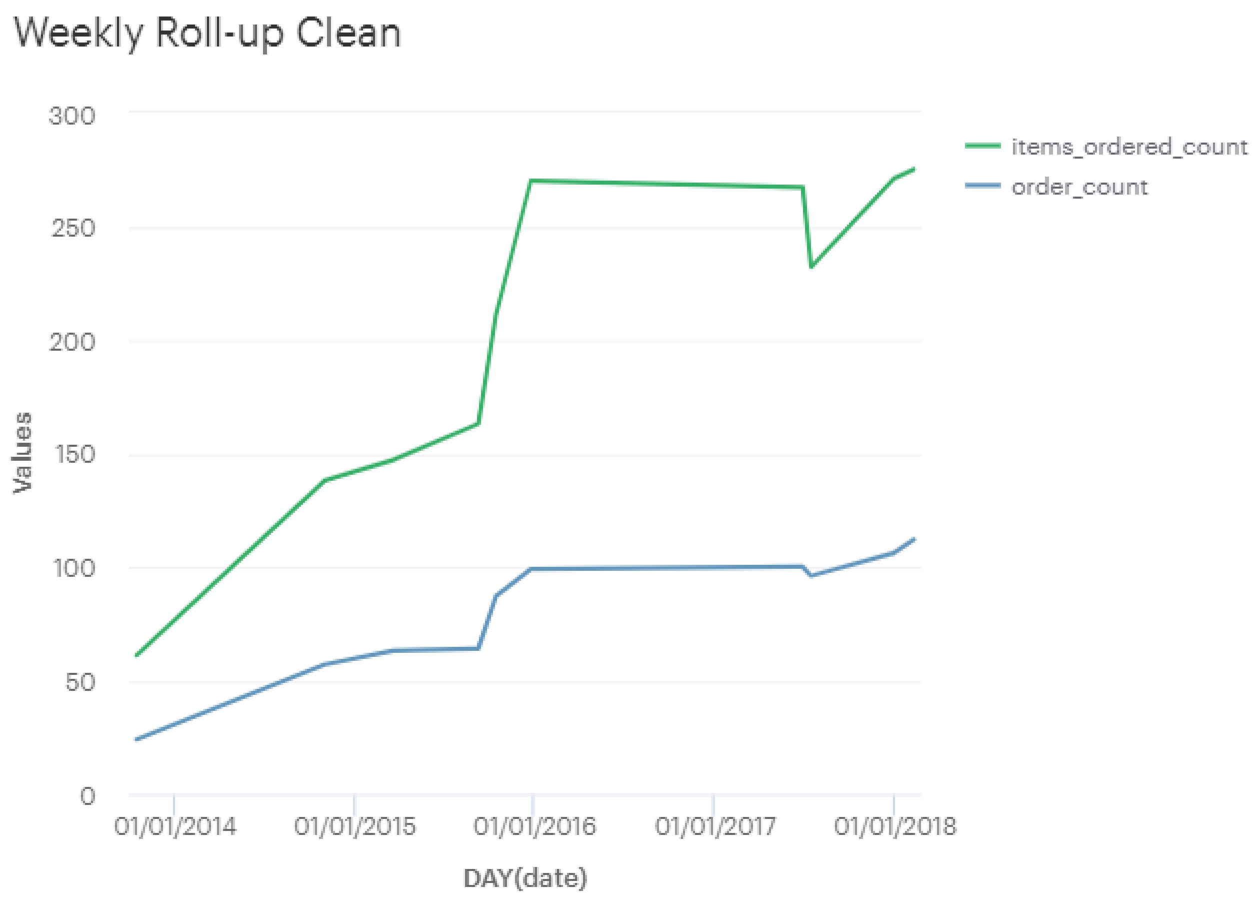
```sql
select *
from
  dsv1069.dates_rollup
left outer join
  (
  select
    date(orders.paid_at)          as day,
    count(distinct invoice_id)    as orders,
    count(distinct line_item_id)  as items_ordered
  from
    dsv1069.orders
  group by
    date(orders.paid_at)
  ) daily_orders
on
  dates_rollup.date >= daily_orders.day
and
  dates_rollup.d7_ago < daily_orders.day
```

Weekly Roll-up



Exercise 5: Column Cleanup. Finish creating the weekly rolling orders table, by performing any aggregation steps and naming your columns appropriately.

```sql
select
  dates_rollup.date,
  sum(orders)         as order_count,
  sum(items_ordered)  as items_ordered_count,
  count(day)          as rows_collapsed
from
  dsv1069.dates_rollup
left outer join
  (
  select
    date(orders.paid_at)          as day,
    count(distinct invoice_id)    as orders,
    count(distinct line_item_id)  as items_ordered
  from
    dsv1069.orders
  group by
    date(orders.paid_at)
  ) daily_orders
on
  dates_rollup.date >= daily_orders.day
and
  dates_rollup.d7_ago < daily_orders.day
group by
  dates_rollup.date
```



Check for anomalies:

```sql
select *
from dsv1069.events
where event_time > '2018-06-02'
```

Looks like your query didn't return any results

Try broadening your query.

Mode Report Link: https://app.mode.com/sum14/reports/2ba17c913105