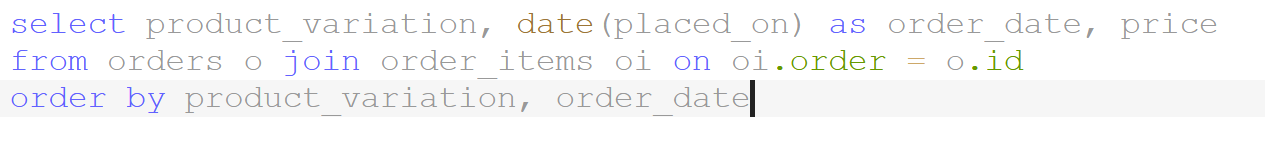
1. Are there incidences of shops are increasing their prices? Does it occur on a regular basis?
2. What is the average annual price increase of products in this database, if any.
3. If it were being redeveloped, what changes would you make to the database schema given to make it more flexible?
4. Does the query (/queries) you wrote scale? What if there were hundreds of thousands of products, customers, variations and orders? What changes might you make to your technique, or the database itself, to optimize this sort of analysis?
5. Without rewriting it, how would your analysis change if the prices were presented in multiple currencies?
6. Based on your findings above, would you recommend building the new feature? Why or why not?

Forked data:

<https://www.db-fiddle.com/f/qj6hWpANHuDYKHfJ9DUuC9/0>

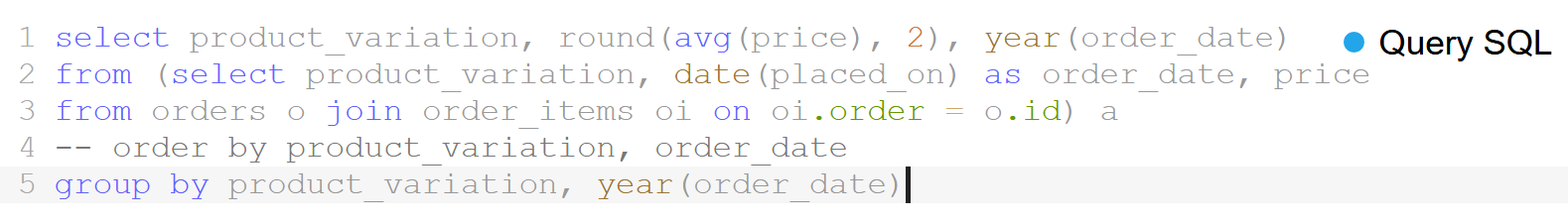
**Q1:**



Query above show each product’s price change history induced from order history.

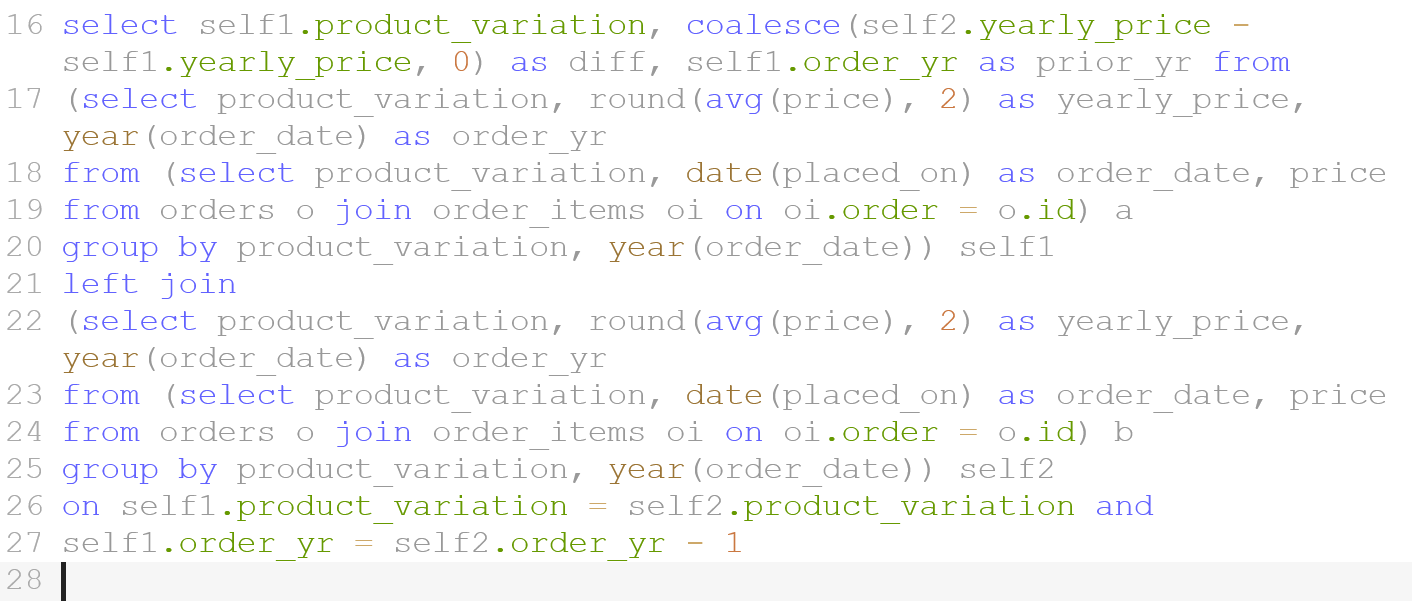
Depending on the product (or unique product\_variation number), the price change, whether increase or decrease can occur in different time scales. Some price changes occur during same day. Others are less often and might only adjust every few weeks or months.

Query below show average yearly price for each product variation.

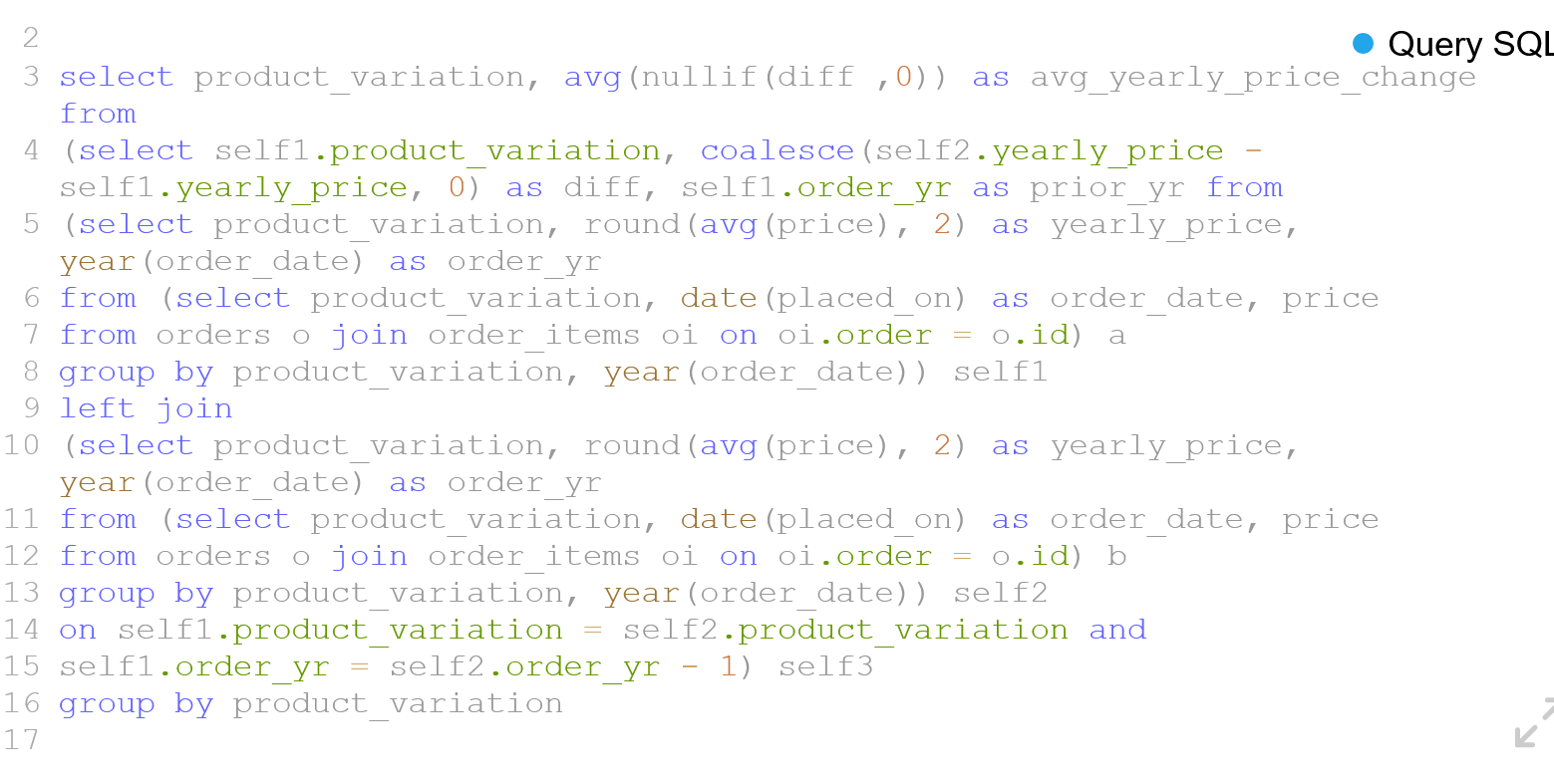


**Q2:**

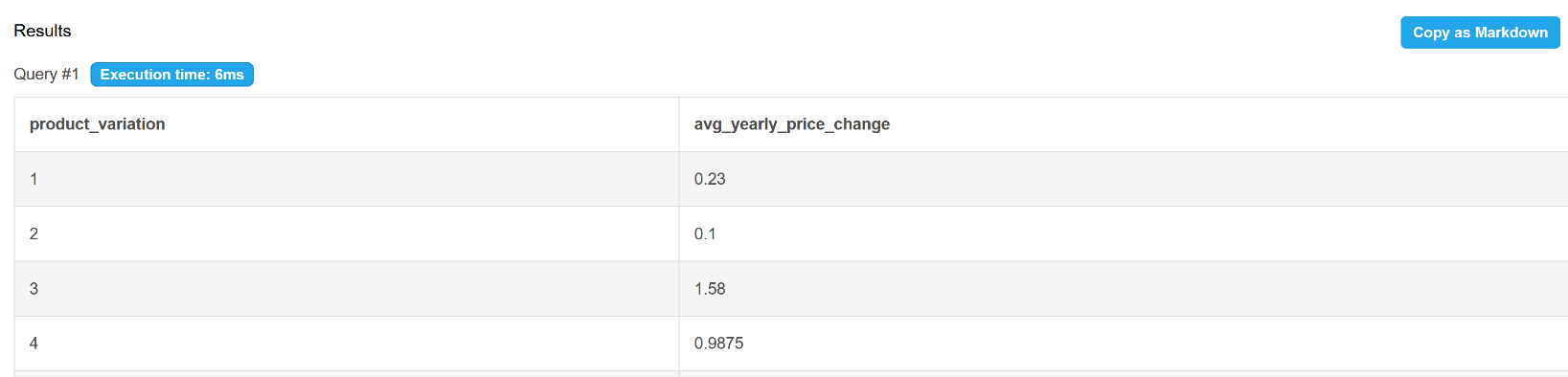
Query below shows change in price from prior year:



Query below shows average yearly price increase for each product



Below are result for first four product variation



Below are text form of the final query:

select product\_variation, avg(nullif(diff ,0)) as avg\_yearly\_price\_change from

(select self1.product\_variation, coalesce(self2.yearly\_price - self1.yearly\_price, 0) as diff, self1.order\_yr as prior\_yr from

(select product\_variation, round(avg(price), 2) as yearly\_price, year(order\_date) as order\_yr

from (select product\_variation, date(placed\_on) as order\_date, price

from orders o join order\_items oi on oi.order = o.id) a

group by product\_variation, year(order\_date)) self1

left join

(select product\_variation, round(avg(price), 2) as yearly\_price, year(order\_date) as order\_yr

from (select product\_variation, date(placed\_on) as order\_date, price

from orders o join order\_items oi on oi.order = o.id) b

group by product\_variation, year(order\_date)) self2

on self1.product\_variation = self2.product\_variation and

self1.order\_yr = self2.order\_yr - 1) self3

group by product\_variation

**Q3:**

To make database schema more flexible and to make query easier, less complex, I suggest including placed\_on timestamp column to order\_items table. This way, joining between orders and order\_items table can be avoided if similar queries are executed in the future.

**Q4:**

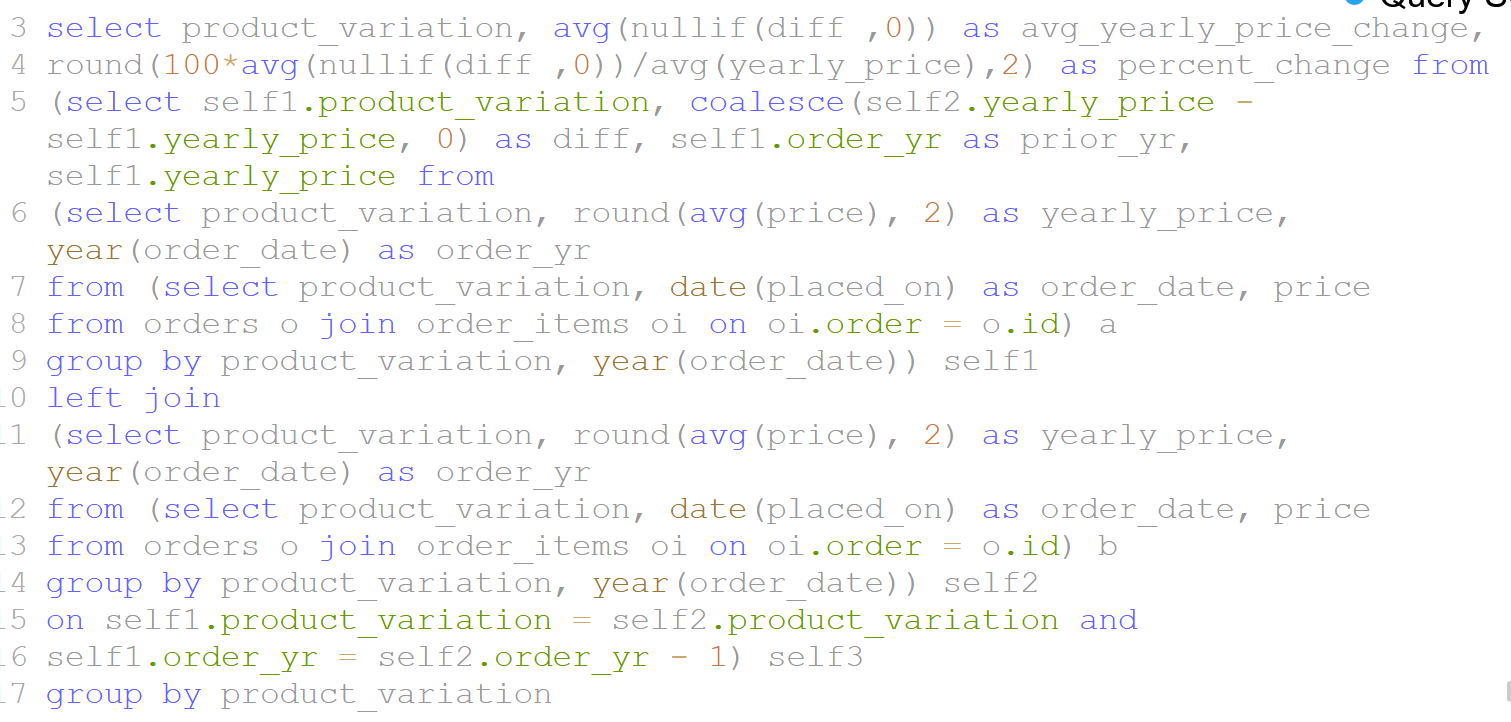
The query I wrote involves two layers of joining so it will not scale well if tables involved have very large amount of rows. To optimize this sort of analysis, I suggest increase redundancy of information stored in the tables. For example, include placed\_on column in order\_items table.

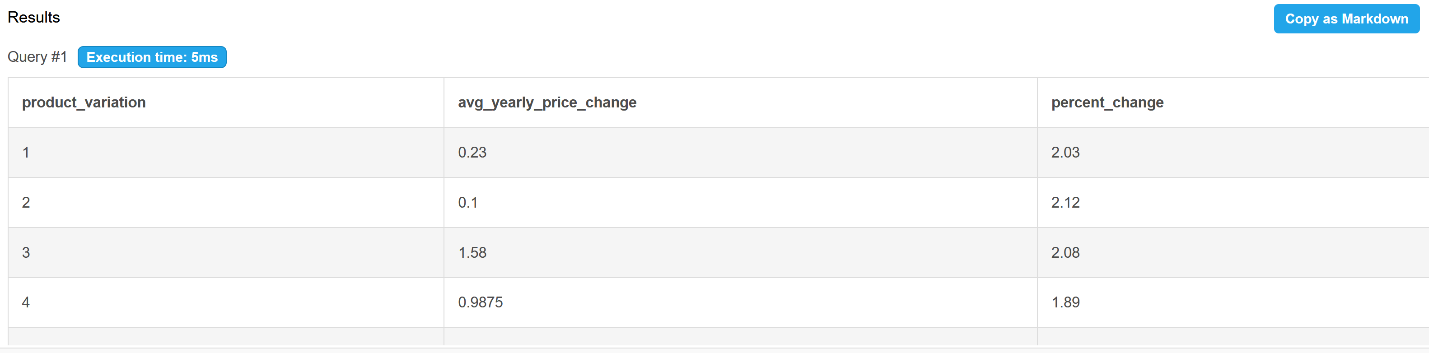
**Q5:**

There are different ways to approach this problem if prices are shown in different currencies. I could only include orders purchased with USD in my analysis. Or I could find out which currency dominates the transaction for each product variation and use that currency for the analysis. I could also convert all currencies to USD using current exchange rate and then perform the same query in Q2.

**Q6:**

Based on Q1 and Q2’s analysis, almost all of products have seen their prices increase year from year. I created another query to derive percentage of price change.





We can observe from the query result above that most products’ price are already being increased at around 2 – 3% YOY, in line with inflation. Based on this data and result from Q1 and Q2, there doesn’t appear to be a need to develop new feature for simplifying price change. However, it’s also possible that current price change function is not good enough but merchants found ways to adapt. Therefore, a survey to merchants is recommended to assess the need of building a new feature for simplifying price change process.

select product\_variation, avg(nullif(diff ,0)) as avg\_yearly\_price\_change,

round(100\*avg(nullif(diff ,0))/avg(yearly\_price),2) as percent\_change from

(select self1.product\_variation, coalesce(self2.yearly\_price - self1.yearly\_price, 0) as diff, self1.order\_yr as prior\_yr, self1.yearly\_price from

(select product\_variation, round(avg(price), 2) as yearly\_price, year(order\_date) as order\_yr

from (select product\_variation, date(placed\_on) as order\_date, price

from orders o join order\_items oi on oi.order = o.id) a

group by product\_variation, year(order\_date)) self1

left join

(select product\_variation, round(avg(price), 2) as yearly\_price, year(order\_date) as order\_yr

from (select product\_variation, date(placed\_on) as order\_date, price

from orders o join order\_items oi on oi.order = o.id) b

group by product\_variation, year(order\_date)) self2

on self1.product\_variation = self2.product\_variation and

self1.order\_yr = self2.order\_yr - 1) self3

group by product\_variation