Data Science Application Challenge- Summer 2019

Background

At Shopify, we're always looking for ways to improve our merchant experience. Recently, someone on one of our teams wondered how often merchants were updating the price of their products, not for temporary sales but as permanent price increases. If this was indeed being done frequently, perhaps we could offer a new feature to make these sorts of changes easier.

Hypothesis

Merchants might have many reasons for occasionally bumping up their prices, such as market conditions commodity prices, but as a minimum it is predicted that price increments should at least reflect overall inflation. If over a long period of time they do not, it is possible that merchants are *not* updating their prices as frequently as they should be. This means we ought to explore building a new feature.

Instructions

Please visit the following link:
https://www.db-fiddle.com/f/svQD6mgBJDcykiJAd4oe8w/7
Before writing any queries, please *fork* this fiddle so you have your own copy of the data.

This web-based mysql database contains mock data for a fake e-commerce platform, not too unlike Shopify's.

As in most e-commerce platforms, there are products as well as product variations. Product variations have the same price as their parent product, but map to different items (such as different colours of the same t-shirt).

Note that this is not an ideal data set. Prices listed in the *products* table reflect current prices; changes merchants have made to these prices have not been recorded in this database. Note that you may not require all of the data provided in order to answer the questions below..

Questions and Answers

Where applicable, please submit the SQL queries and any assumptions made along with your responses to these questions.

Preface

Before being able to determine the answers to the questions below, I must first get myself acquainted with the database set up. I want to learn the main tables that are related to price data, the types of data stored in these tables, and how these tables are connected. To do this I ran the following queries:

```
select * from product
```

With this query I determined that I would not be able to find any changes in pricing directly from the product table.

```
select * from orders
```

With this query I learned that orders are recorded with respect to product_variations, so I need to look there next

```
select * from product_variations order by product
```

Here I am able to see that there are in fact multiple product variations for many products. I can also see that the names of products can be null or empty. Of course I could have learned all of this from looking at the schema, but for a large database reading the schema could be overly time consuming for only a short task like this.

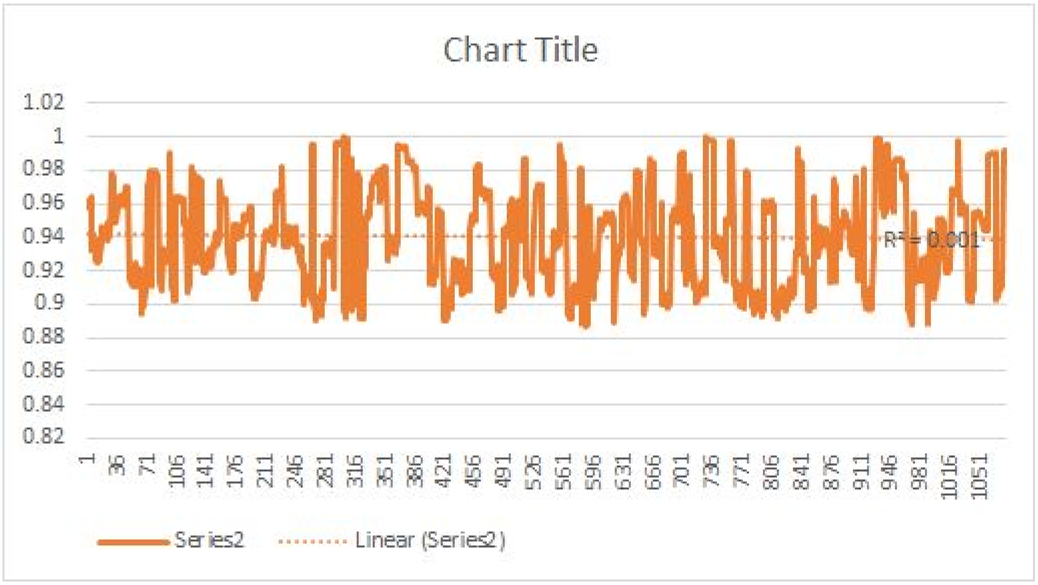
```
select * from order_items
```

This table looks like it is going to be the most useful. It gives me the price of each product_variation, and from the orders table I can get the date that each item was priced this way at.

```
select * from product_variations join products on (products.id =
product_variations.`product`)
```

```
select order_items.`order`, order_items.price / products.price from order_items join
product_variations  on(`order_items`.product_variation = product_variations.id) join
products on (products.id = product_variations.`product`)
```

This query tells us the ratio of price between the price when it was ordered and the current price of the product. A value of 1 means the price has not changed. A value below 1 means the price has gone up, and a price above 1 means the price has gone down. A quick look through the results for this query seem to indicate that prices went down and then went up, but are actually still below current prices.

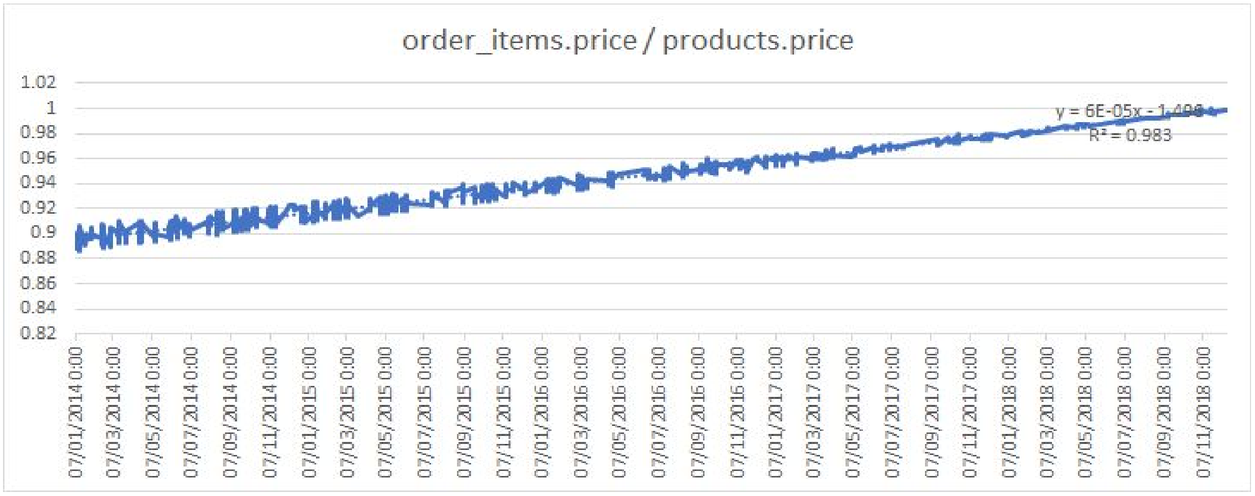


I copy and pasted the results into excel to create the above chart. From this chart we can see that prices have been changing regularly, but prices have not just been going up. They seem to go up and down over the course of transactions. Next we should change our x-axis to see if there is a trend over time that is not being taken into account by our ordering by order-number.

select order_items.`order`, orders.placed_on, order_items.price / products.price from order_items join product_variations on(`order_items`.product_variation = product_variations.id) join products on (products.id = product_variations.`product`) join `orders` on(`orders`.id = order_items.`order`) order by orders.placed_on

Using this query I was once again able to construct a graph in Excel. I would just like to note that copying and pasting data into excel is not my preferred method of visualizing data. When I worked at NCR as an intern over the summer I developed an automated system to export data from an SQL server database to excel using ODBC, and I developed some excel macros to automatically transform the data into a presentable format. If I were doing more than just data mining here, I would likely want to create a system like that.

1. Are there incidences of shops are increasing their prices? Does it occur on a regular basis?



Based on the above graph we can clearly see a trend of increasing prices over time. According to excel there is a linear trend with equation (approximately) y = 0.0000574x - 1.49 an R^2 value of 0.983 indicating a relatively high degree of correlation. There is also an exponential trend with equation $y = 0.0711e^{6E\text{-}05x}$ and R² = 0.9818, though given that our x value is actually in discrete time instead of continuous time, this may not actually represent the true equation. I mention the exponential trend here because we would expect prices to increase exponentially as inflation does.

2. What is the average annual price increase of products in this database, if any.

We will use the average annual growth rate formula with data from the linear trend line we got from excel (=0.0000574317451415184*DATEVALUE("01/1/2015")-1.49597648005271)

Year	Date	Normalized Price	Growth
2014	01/01/2014	0.895481388	
2015	01/01/2015	0.916443975	2.34%
2016	01/01/2016	0.937406562	2.28%
2017	01/01/2017	0.95842658	2.24%
2018	01/01/2018	0.979389167	2.18%
2019	01/01/2018	1.000351754	2.14%

Therefore, the average annualized growth is roughly **2.24%.** Note, this is not compounding growth, though we may want to calculate that as well.

3. If it were being redeveloped, what changes would you make to the database schema given to make it more flexible?

Maybe I am missing something, but this database scheme seems to be a very flexible design, as there are very few limitations built into this database. I would actually prefer a less flexible model which implements foreign keys, as well as some other data consistency checks such as making sure SKUs and emails are valid. Additionally I would make some minor changes to the typing of certain attributes, such as changing price from a float to a more consistent format (either an int multiplied by 100, two ints representing dollars and cents respectively, or a built in currency type if available). As mentioned in a latter question, it could be worthwhile to add a currency type attribute wherever price shows up, or to the merchant table.

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?

All of the queries I wrote are reasonably efficient as they use joins instead of less efficient methods of combining tables. Overall I would make sure that the large tables I am searching are properly indexed, and that I am not performing these queries while the database is under heavy load from another source. Usually if I were faced with a database optimisation problem, I would start by using any database tools I have at my disposal (such as those that exist in MS SQL server). If I found that the query I wrote was taking too much time, I would use the Explain keyword to narrow down the bottleneck. If this data were stored on a production database, I would consider transferring it to a test or other non production database where using a lot of resources would be less of a concern.

5. Without rewriting it, how would your analysis change if the prices were presented in multiple currencies?

As long as the currency remained the same for each item (ie. items priced in dollars are always priced in dollars) this would not cause any problems because our prices are normalized by dividing past prices by the current price. If this were not true, and items could have had their currencies changed without the item changing IDs, we would not be able to use the above method to calculate price increases, and would have to include some exchange rate information in our calculations.

6. Based on your findings above, would you recommend building the new feature? Why or why not?

It certainly seems that merchants are increasing their prices over time, and as hypothesized the average rate of increase roughly matches inflation. It seems that a feature to improve the ease of which merchants increase prices would meet a need, and that such a feature would be relatively inexpensive to implement. Additionally, other new features could be built on top of this, such as price recommendations. Such recommendations could be built by looking at price data across the market (to gage inflation) as well as by individual sectors to give a more highly tailored answer. An effective price prediction feature could be highly profitable to merchants, and would be good for society as it would increase the efficiency of the market as a whole.

Currently, only medium and large corporations have the budgets to be able to hire data science professionals, but a feature like this would allow even small business to take advantage of