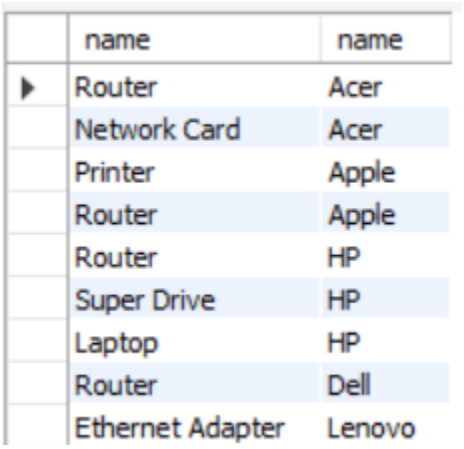
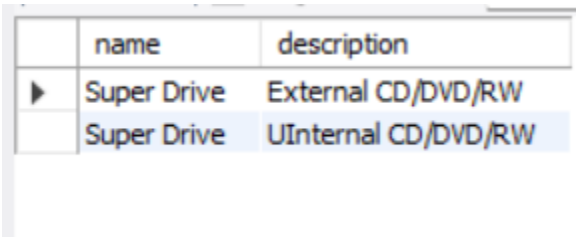
Author: Joseph Pepe
Assignment: HW3

Query1:
First we will select the product name and the company. Then, we need to then join from products -> sell -> merchants, and just tacking a where clause on the end where sell_quantity is 0 will get us the answer.



name	name
Router	Acer
Network Card	Acer
Printer	Apple
Router	Apple
Router	HP
Super Drive	HP
Laptop	HP
Router	Dell
Ethernet Adapter	Lenovo

Query2:
We will select the names and descriptions of products, and we need to join sell to see if anything is sold, and we include a where clause because we want products that aren"t sold, or null.



name	description
Super Drive	External CD/DVD/RW
Super Drive	UInternal CD/DVD/RW

Query3:
We want to select customers and the number of orders that we will filter later. We join place -> orders -> contain -> products in order to be able to filter by the name where the name has Sata. We will make a subquery in the and clause because we also want people who did not by routers, so we will do a join from customers all the way over to products again and select people who bought routers. In the main query, they won't be selected though because of the NOT.

	fullname	COUNT(orders.oid)

Query4:
This question was left up to interpretation. I chose to select product name and sell price, and display the items at a 20% discount. In order to do this, I needed to go from products -> sell -> merchants, so I could filter by 'HP' and 'Networking' to make sure I am only selecting what the problem wants.

	name	ROUND(sell.price * 0.8, 2)
▶	Router	827.57
	Network Card	923.74
	Network Card	276.01
	Network Card	209.76
	Ethernet Adapter	1008.36
	Router	164.45
	Router	1179.9
	Router	441.62
	Router	80.76
	Network Card	943.21

Query5:
First, I select the product name, and the min price. There is some tricky stuff going on in the sell table - multiple
merchants sell the same product. This can lead to redundancy in the results if you are not careful. So, I will assume
that Uriel bought the cheapest of each item. Next, I must join place -> contain -> products -> sell then group by product
name and fulter out all customers but Uriel Whitney.

name	price
Monitor	252.01
Router	100.95
Super Drive	61.84
Printer	310.83
Network Card	91.65
Hard Drive	168.13
Laptop	33.5
Ethernet Adapter	126.82
Desktop	311.06

Query6:

I select merchant name, the date (you can use the function year to have SQL order the datetimes into years), and summation of sell price. I need to join customers to place to contain to products to sell to merchants. to get over to merchant names. Now this problem is a little tricky because of the way the schema is - the answer may not be 100% accurate. Becuase of the way contains is, it is not possible to recall which merchant fulfilled which order - therefore it is not possible to recall exact prices because different merchants sell the same items at different prices. So in this table, when I am joining sell, I do so with a subquery that bases it on a consistent price. Finally, I group by name and year.

name	eachYear	revenue
Acer	2011	18643.51
Apple	2011	15930.64
Dell	2011	13180.45
HP	2011	25877.17
Lenovo	2011	7499.04
Acer	2016	6074.84
Apple	2016	6669.08
Dell	2016	5384.51
HP	2016	11680.11
Lenovo	2016	3728.67
Acer	2017	15378.31
Apple	2017	20222.19
Dell	2017	10731.53
HP	2017	30627.24

Query7:

This query is almost exactly the same as the previous one. Again, we select the name of the merchant, year, and the revenue. We need to go from customer to merchants so we do customers -> place -> contain -> products -> sell -> merchants. Similar to the last problem, we have this weird issue with the schema where we cannot tell who fulfilled what, so I will make the same assumption. When I am joining sell, I wil do so with a subquery selecting only the min price so that we do not have loads of redundant values when joining the tables. The main difference is that we are ordering by revenue instead of company and year and then limiting it to 1.

	name	eachYear	revenue
▶	Acer	2018	42191.17

Query8:
This query is relatively simple. We can select the shipping methods and get the average of the shipping cost, then group by method and order it in ascending order so the cheapest is displayed.

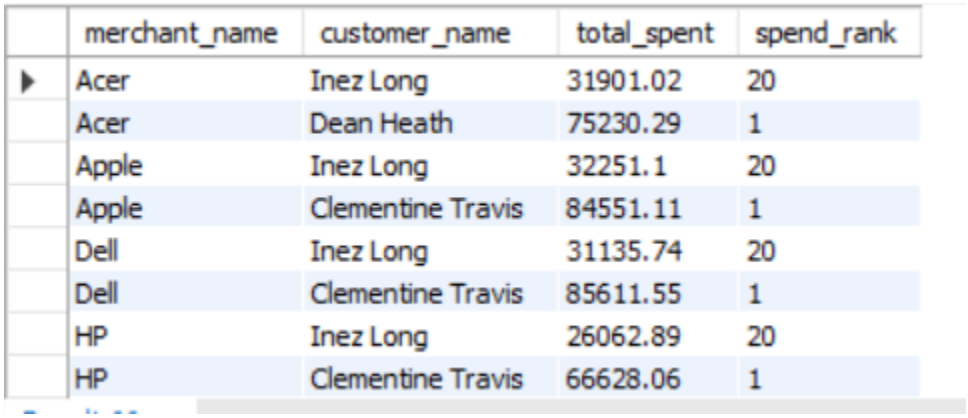
	shipping_method	shipcost
▶	USPS	7.46

Query9:
First I will select merchant name, product category, and the revenue. I need to go from sell -> products -> merchants in order to include the merchant name. While joining I am able to get the product category and I can calculate price from the sell table. Finally, I just group my merchant name and category and order it by sales.

	merchant_name	category	total_sales
▶	Acer	Peripheral	11656.7
	Apple	Peripheral	11358.03
	Lenovo	Peripheral	11037.42
	Dell	Peripheral	10816.99
	Lenovo	Networking	8803.21
	Apple	Networking	7774.73
	HP	Networking	7569.21
	Dell	Networking	7376.57

Query10:
I start by selecting the name, customer, the amount the customer spent, and their rank. Next, we open a subquery. It was a little difficult
to figure out how to rank by how much they spent - it was hard to get the number to display for both highest and lowest at once. Therefore,

I did some researching and we are able to use a subquery and this RANK() function. Here is what I used to learn about it: https://www.geeksforgeeks.org/sql/rank-function-in-sql-server/
The inner subquery is basically just getting the total spending per customer per merchant, and that is done by selecting the customer names, getting the total amount that they spent, and assigning a rank to them. We need to just join all of the tables to be able to connect customer to merchants. The outer query is quite simple, it just filters to make sure that only the highest and lowest are displayed, and since we are displaying the merchants, the query will return the highest and lowest.



merchant_name	customer_name	total_spent	spend_rank
Acer	Inez Long	31901.02	20
Acer	Dean Heath	75230.29	1
Apple	Inez Long	32251.1	20
Apple	Clementine Travis	84551.11	1
Dell	Inez Long	31135.74	20
Dell	Clementine Travis	85611.55	1
HP	Inez Long	26062.89	20
HP	Clementine Travis	66628.06	1