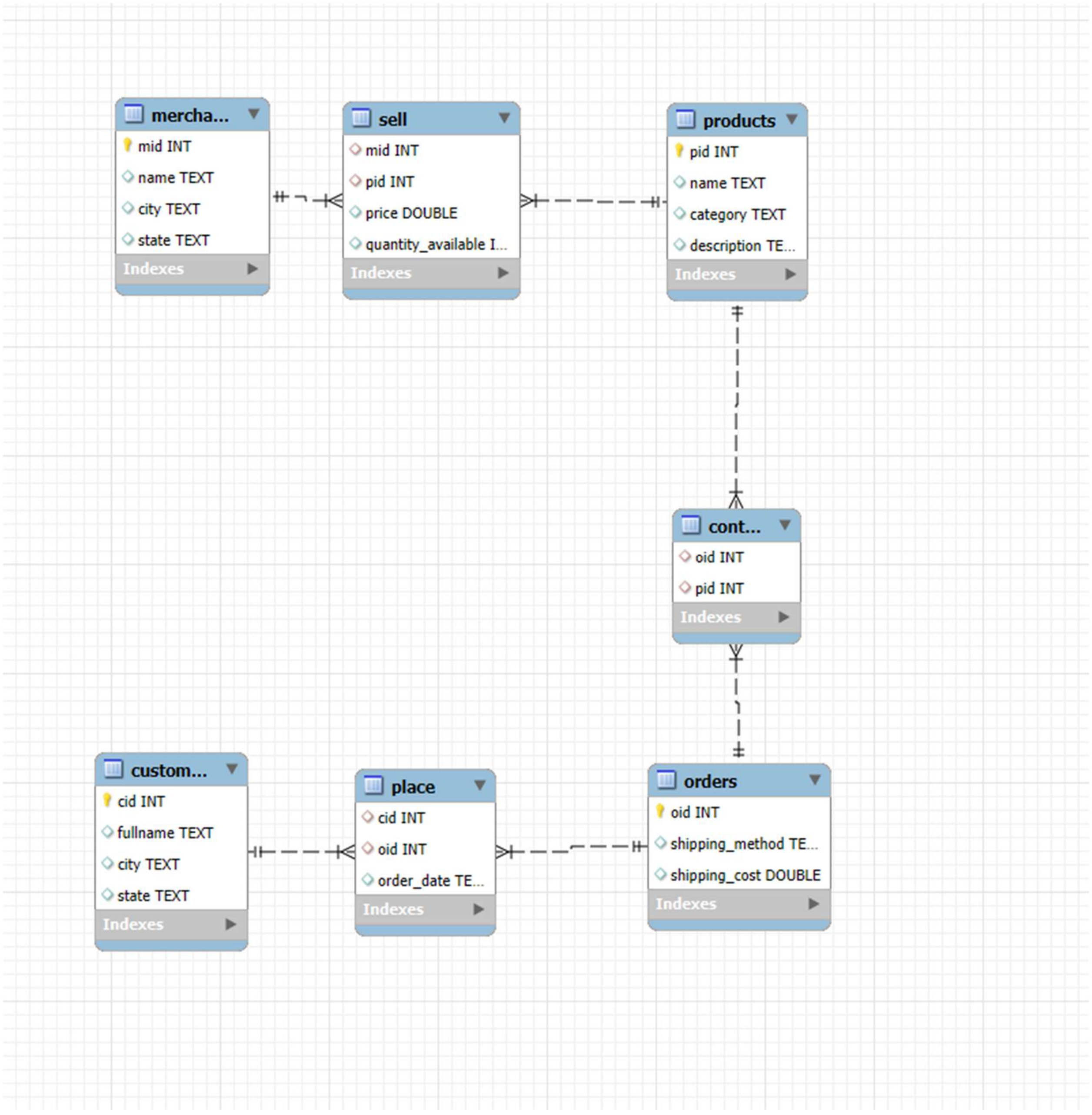
Title: DB Assignment 3

Name: Paul Anderson

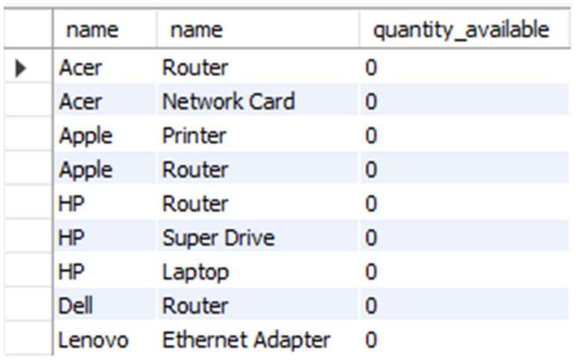
Date: 19 October 2025

ERD Diagram



Query 1: List names and sellers of products that are no longer available

```
--  simply finding products that have a quantity of 0, implying they have sold out or are nolonger available

select merchants.name, products.name, sell.quantity_available

from merchants

join sell on merchants.mid = sell.mid

join products on products.pid = sell.pid

where quantity_available = 0;
```



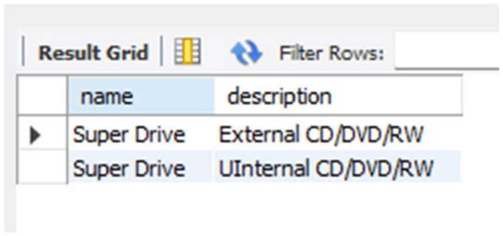
name	name	quantity_available
Acer	Router	0
Acer	Network Card	0
Apple	Printer	0
Apple	Router	0
HP	Router	0
HP	Super Drive	0
HP	Laptop	0
Dell	Router	0
Lenovo	Ethernet Adapter	0

Explanation: All products that have 0 quantity available are not currently being sold.

Query 2: List names and descriptions of products thare are not sold

```
-- getting a list of all products

select products.name, products.description

from products


except


-- getting all products that have been sold

select products.name, products.description
```

from products natural join sell;



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

Explanation: the first query gets all of the products available, then uses the except keyword to subtract a query of all products that can be joined with sell. Essentially it is returning the values that are have a null value when referenced with the sell table.

Query 3: How many customers bought SATA drives but not any routers

-- Getting everyone who bought an SSD

select products.name, count(customers.cid) as purchase_count

from products

join contain

on products.pid = contain.pid

join orders

on contain.oid = orders.oid

join place

on orders.oid = place.oid

join customers on place.cid = customers.cid

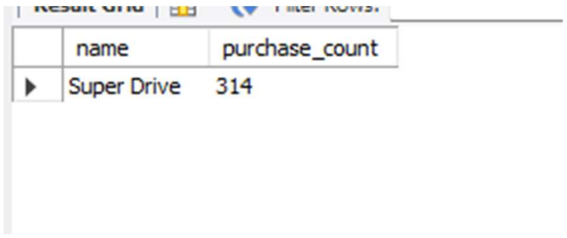
group by products.name

having products.name = 'Super Drive'

-- Leaving out

except

```
-- Getting everyone who purchased a router
select products.name, count(customers.cid) as purchase_count
from products
join contain
on products.pid = contain.pid
join orders
on contain.oid = orders.oid
join place
on orders.oid = place.oid
join customers on place.cid = customers.cid
group by products.name
having products.name = 'Routers'
```



name	purchase_count
Super Drive	314

Explanation: This is doing something similar to query 2. It is getting a count of all the customers that purchased an SSD (super drive) than using the except keyword to remove everyone who also bought a router.

Query 4: HP has 20% sale on all its networking products

```
-- applying a 20% discount by mutliplcation of sell.price by 0.8
```

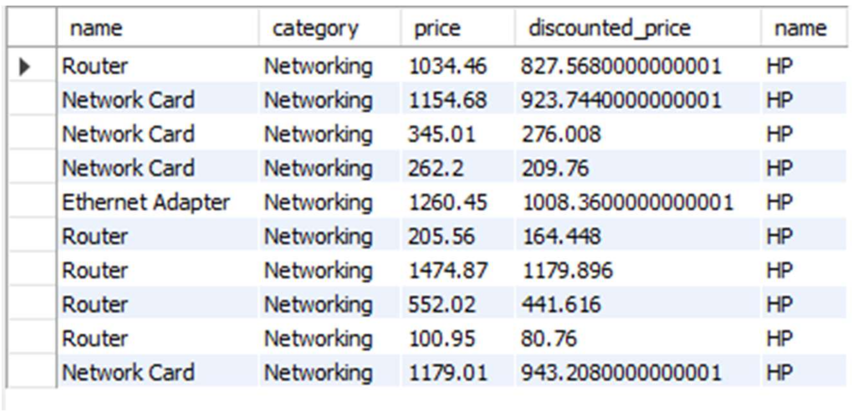
select products.name, products.category, sell.price, sell.price * .8 as discounted_price, merchants.name

from products

join sell on products.pid = sell.pid

join merchants on sell.mid = merchants.mid

where merchants.name = 'HP' and products.category = 'Networking';



name	category	price	discounted_price	name
Router	Networking	1034.46	827.5680000000001	HP
Network Card	Networking	1154.68	923.7440000000001	HP
Network Card	Networking	345.01	276.008	HP
Network Card	Networking	262.2	209.76	HP
Ethernet Adapter	Networking	1260.45	1008.3600000000001	HP
Router	Networking	205.56	164.448	HP
Router	Networking	1474.87	1179.896	HP
Router	Networking	552.02	441.616	HP
Router	Networking	100.95	80.76	HP
Network Card	Networking	1179.01	943.2080000000001	HP

Explanation: getting everything sold by HP and multiplying it by 0.8 to represent an 20% off discount.

Query 5: What did Uriel Whitney Order (get product name and price)

-- getting everything Uriel ordered

select customers.fullname, products.name, sell.price

from customers

join place on customers.cid = place.cid

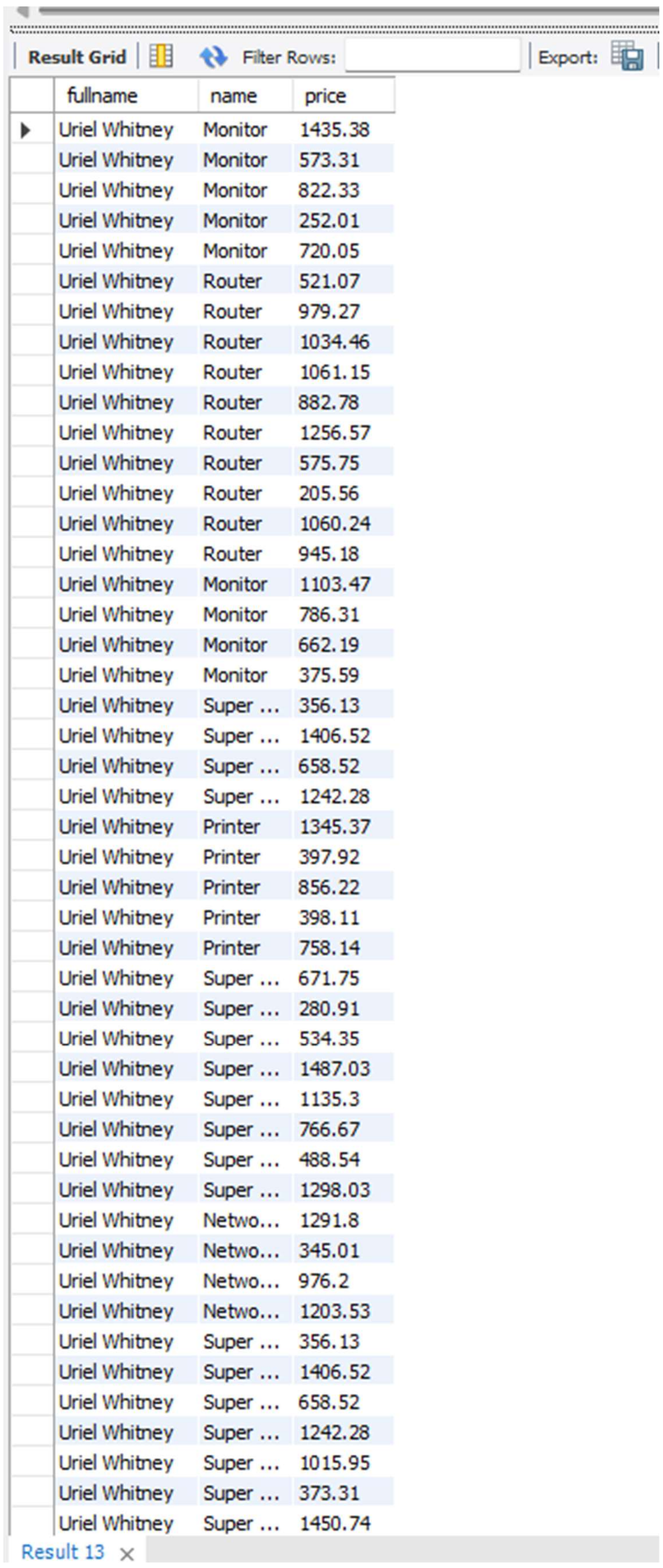
join orders on place.oid = orders.oid

join contain on orders.oid = contain.oid

join products on contain.pid = products.pid

join sell on products.pid = sell.pid

where customers.fullname = 'Uriel Whitney';



Result Grid | Filter Rows: | Export:

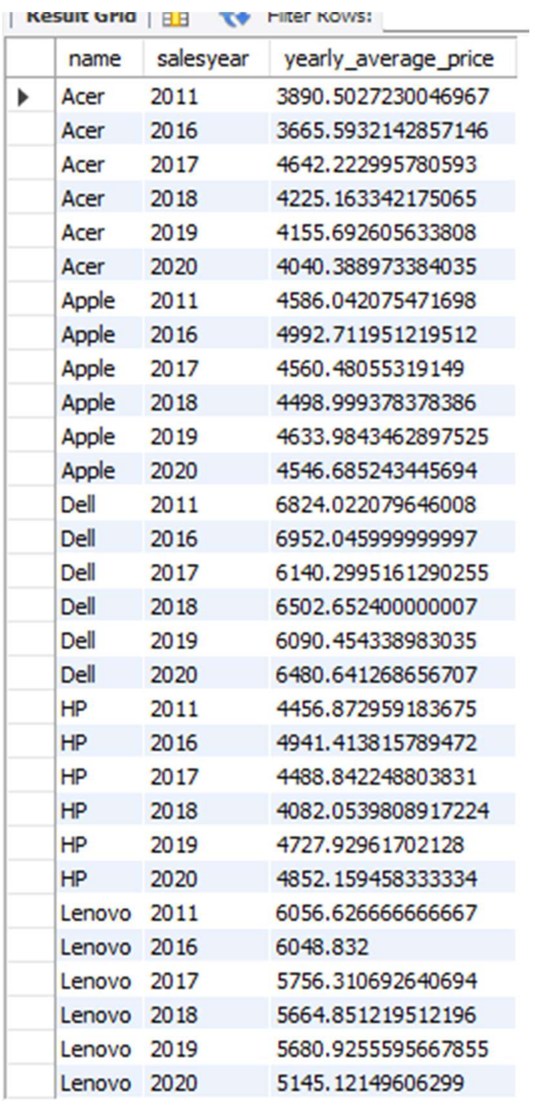
fullname	name	price
Uriel Whitney	Monitor	1435.38
Uriel Whitney	Monitor	573.31
Uriel Whitney	Monitor	822.33
Uriel Whitney	Monitor	252.01
Uriel Whitney	Monitor	720.05
Uriel Whitney	Router	521.07
Uriel Whitney	Router	979.27
Uriel Whitney	Router	1034.46
Uriel Whitney	Router	1061.15
Uriel Whitney	Router	882.78
Uriel Whitney	Router	1256.57
Uriel Whitney	Router	575.75
Uriel Whitney	Router	205.56
Uriel Whitney	Router	1060.24
Uriel Whitney	Router	945.18
Uriel Whitney	Monitor	1103.47
Uriel Whitney	Monitor	786.31
Uriel Whitney	Monitor	662.19
Uriel Whitney	Monitor	375.59
Uriel Whitney	Super ...	356.13
Uriel Whitney	Super ...	1406.52
Uriel Whitney	Super ...	658.52
Uriel Whitney	Super ...	1242.28
Uriel Whitney	Printer	1345.37
Uriel Whitney	Printer	397.92
Uriel Whitney	Printer	856.22
Uriel Whitney	Printer	398.11
Uriel Whitney	Printer	758.14
Uriel Whitney	Super ...	671.75
Uriel Whitney	Super ...	280.91
Uriel Whitney	Super ...	534.35
Uriel Whitney	Super ...	1487.03
Uriel Whitney	Super ...	1135.3
Uriel Whitney	Super ...	766.67
Uriel Whitney	Super ...	488.54
Uriel Whitney	Super ...	1298.03
Uriel Whitney	Netwo...	1291.8
Uriel Whitney	Netwo...	345.01
Uriel Whitney	Netwo...	976.2
Uriel Whitney	Netwo...	1203.53
Uriel Whitney	Super ...	356.13
Uriel Whitney	Super ...	1406.52
Uriel Whitney	Super ...	658.52
Uriel Whitney	Super ...	1242.28
Uriel Whitney	Super ...	1015.95
Uriel Whitney	Super ...	373.31
Uriel Whitney	Super ...	1450.74

Result 13 ×

Explanation: This query got everything Uriel Whitney purchased, the item name, and price of item.

Query 6: List the annual total sales for each company (sort along company and year attribute)

```sql
-- Extracting the year from order_date and then finding the average annual revenue

select merchants.name, extract(year from order_date) as salesyear, avg(price * quantity_available) as yearly_average_price

from merchants

join sell on merchants.mid = sell.mid

join contain on sell.pid = contain.pid

join orders on contain.oid = orders.oid

join place on orders.oid = place.oid

group by merchants.name, salesyear

order by merchants.name, salesyear;
```



name	salesyear	yearly_average_price
Acer	2011	3890.5027230046967
Acer	2016	3665.5932142857146
Acer	2017	4642.222995780593
Acer	2018	4225.163342175065
Acer	2019	4155.692605633808
Acer	2020	4040.388973384035
Apple	2011	4586.042075471698
Apple	2016	4992.711951219512
Apple	2017	4560.48055319149
Apple	2018	4498.999378378386
Apple	2019	4633.9843462897525
Apple	2020	4546.685243445694
Dell	2011	6824.022079646008
Dell	2016	6952.045999999997
Dell	2017	6140.2995161290255
Dell	2018	6502.652400000007
Dell	2019	6090.454338983035
Dell	2020	6480.641268656707
HP	2011	4456.872959183675
HP	2016	4941.413815789472
HP	2017	4488.842248803831
HP	2018	4082.0539808917224
HP	2019	4727.92961702128
HP	2020	4852.159458333334
Lenovo	2011	6056.626666666667
Lenovo	2016	6048.832
Lenovo	2017	5756.310692640694
Lenovo	2018	5664.851219512196
Lenovo	2019	5680.9255595667855
Lenovo	2020	5145.12149606299

Explanation: This query is getting every companies average spending per year by using the extract key work and extracting the year , ordered by company. Strangely, there is a gap between 2011 and 2016 for every single company.

Query 7: Which company has the highest annual revenue & what year

-- Extracting the year from order_date and then finding the highest annual revenue

select merchants.name, extract(year from order_date) as salesyear, avg(price * quantity_available) as yearly_average_price

from merchants

join sell on merchants.mid = sell.mid

join contain on sell.pid = contain.pid

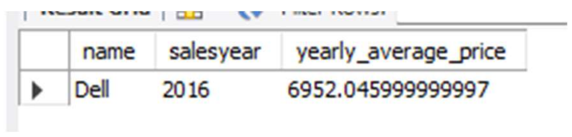
join orders on contain.oid = orders.oid

join place on orders.oid = place.oid

group by merchants.name, salesyear

order by yearly_average_price desc

limit 1;



	name	salesyear	yearly_average_price
▶	Dell	2016	6952.045999999997

Explanation: Getting the sale year by using the extract key work, getting the yearly average price. Then ordering by the price instead of the company. Limiting to 1 to get the top company.

Query 8: on average what was the cheapest shipping method available

-- relatively simple query, just getting the average_shipping_cost and just finding the cheapest one

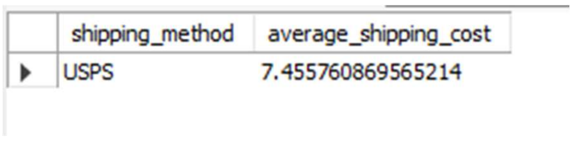
select shipping_method, avg(shipping_cost) as average_shipping_cost

from orders

group by shipping_method

order by average_shipping_cost asc

limit 1;

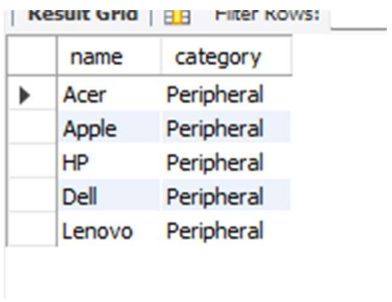


	shipping_method	average_shipping_cost
▶	USPS	7.455760869565214

Explanation: Getting the average shipping cost and shipping method, ordering by shipping cost ascending, and limiting to 1 to get the minimum.

Query 9: What is the best sold ($) category for each company

```
-- splitting up this query into multiple parts to make it easier to read.
with amount_sold as (
        select products.pid, products.category, count(orders.oid) as num_orders
        from products
         join contain on products.pid = contain.pid
         join orders on contain.oid = orders.oid
        group by products.pid, products.category
         having num_orders >= all (
                select count(orders.oid)
                from products
                join contain on products.pid = contain.pid
                join orders on contain.oid = orders.oid
                group by products.pid, products.category
  )
)

select merchants.name, amount_sold.category
from merchants
join sell on merchants.mid = sell.mid
join amount_sold on sell.pid = amount_sold.pid
order by amount_sold.num_orders desc;
```


Result Grid | Filter Rows:

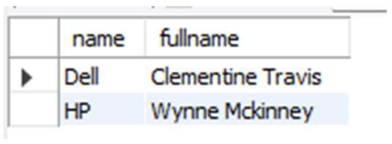
name	category
Acer	Peripheral
Apple	Peripheral
HP	Peripheral
Dell	Peripheral
Lenovo	Peripheral

Explanation: CTE to split up the query into more manageable chunks. First one is getting the amount of each category sold, and finally ordering by the amount sold descending.

Query 10: For each company find out which customers have spent the most and the least amounts

```
-- CTE to find most and least spender

-- this subquery find the most spending and least spending customer

with most_spent as (

        select merchants.name, customers.fullname, sum(sell.price *
sell.quantity_available) as max_total_spent

  from merchants

  join sell on merchants.mid = sell.mid

  join products on sell.pid = products.pid

  join contain on products.pid = contain.pid

  join orders on contain.oid = orders.oid

  join place on orders.oid = place.oid

  join customers on place.cid = customers.cid

  group by merchants.name , customers.fullname

  having max_total_spent >= all ( -- searching for the most spending

                select sum(sell.price * sell.quantity_available)

                from merchants
```

```
                    join sell on merchants.mid = sell.mid
                    join products on sell.pid = products.pid
                    join contain on products.pid = contain.pid
                    join orders on contain.oid = orders.oid
                    join place on orders.oid = place.oid
                    join customers on place.cid = customers.cid
      group by merchants.name, customers.fullname
  ) or max_total_spent <= all ( -- searching for the least spending
                    select sum(sell.price * sell.quantity_available)
                    from merchants
                    join sell on merchants.mid = sell.mid
                    join products on sell.pid = products.pid
                    join contain on products.pid = contain.pid
                    join orders on contain.oid = orders.oid
                    join place on orders.oid = place.oid
                    join customers on place.cid = customers.cid
      group by merchants.name, customers.fullname
  )
  order by merchants.name, customers.fullname
)

-- main query
select most_spent.name, most_spent.fullname
from most_spent
group by most_spent.name, most_spent.fullname
```



name	fullname
Dell	Clementine Travis
HP	Wynne Mckinney

Explanation:

Using a CTE to search for each company and getting their least and most spending customer. It was not working to get more than the top and bottom merchant/customer combination.