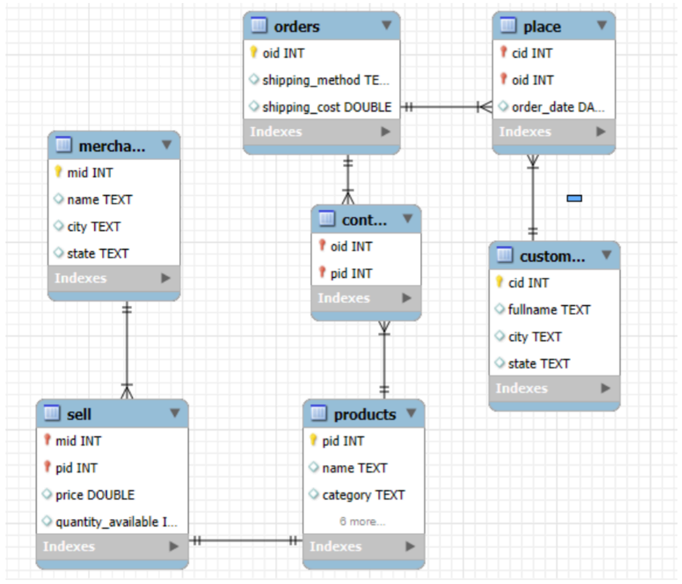
DB Assignment 3

Nick Kraus

10/21/25

- ## ER Diagram



- ## Primary and Foreign Keys

```
-- Adding primary and composite keys for imported data
alter table merchants add primary key (mid);
alter table products add primary key (pid);
alter table sell add primary key (mid, pid);
alter table orders add primary key (oid);
alter table contain add primary key (oid, pid);
alter table customers add primary key (cid);
alter table place add primary key (cid, oid);

-- Adding foreign keys
alter table sell add foreign key (mid) references merchants (mid),
  add foreign key (pid) references products (pid);
alter table contain add foreign key (oid) references orders (oid),
  add foreign key (pid) references products (pid);
alter table place add foreign key (cid) references customers (cid),
  add foreign key (oid) references orders (oid);
```

- **Constraints**

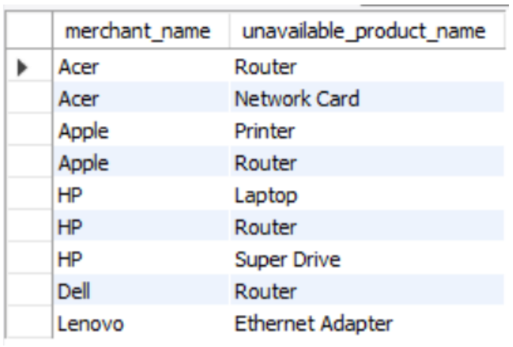
```
-- Valid product name constraint
alter table products
  add constraint valid_name
  check (name in ('Printer','Ethernet Adapter','Desktop','Hard Drive','Laptop','Router','Network Card','Super Drive','Monitor'));

-- Valid product category constraint
alter table products
  add constraint valid_product_category
  check (category in ('Peripheral', 'Networking', 'Computer'));

-- Valid sale price constraint
alter table sell
  add constraint valid_price
  check (price between 0 and 100000);

-- Valid quantity of available product constraint
alter table sell
  add constraint valid_quantity_available
  check (quantity_available between 0 and 1000);
```

```
-- Valid shipping method constraint
alter table orders
  add constraint valid_shipping_method
  check (shipping_method in ('UPS', 'FedEx', 'USPS'));

-- Valid shipping cost constraint
alter table orders
  add constraint valid_shipping_cost
  check (shipping_cost between 0 and 500);

-- Valid date constraint
alter table place
  modify column order_date date;
```

1. **List names and sellers of products that are no longer available (quantity=0)**

```
-- #1 Prints table of the sellers names and their products that have a zero products available by
-- joining sell, merchant, and product
select merchants.name as merchant_name, products.name as unavailable_product_name
from sell s
join merchants using (mid)
join products using (pid)
where s.quantity_available = 0;
```

This query searches through tables sell, merchants, and products using a join using method. It then uses a where clause to find products that have a quantity_available value of 0.

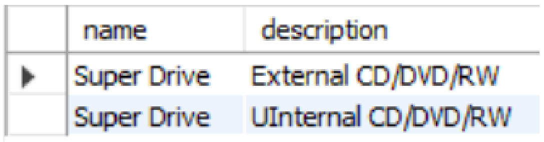


merchant_name	unavailable_product_name
Acer	Router
Acer	Network Card
Apple	Printer
Apple	Router
HP	Laptop
HP	Router
HP	Super Drive
Dell	Router
Lenovo	Ethernet Adapter

2. **List names and descriptions of products that are not sold.**

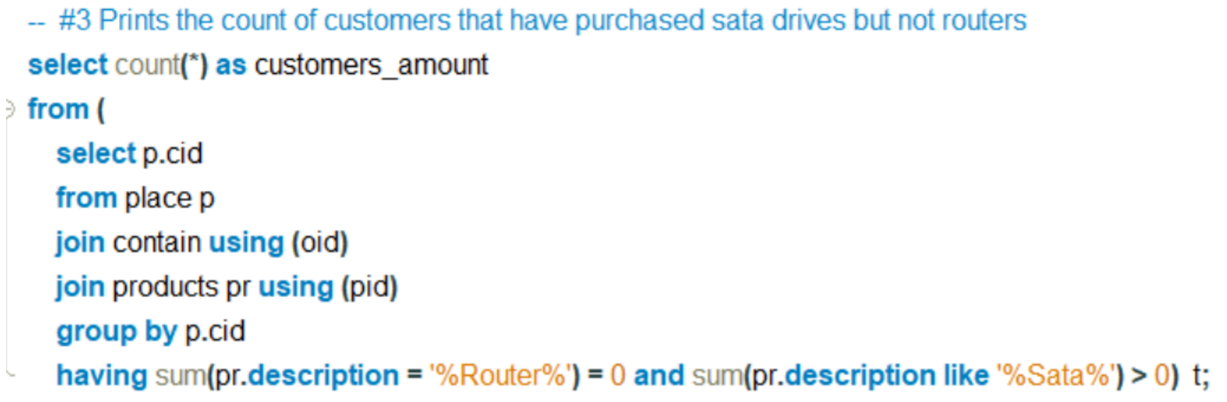
```
-- #2 Prints a table of products' name and description that aren't being sold
select p.name, description
from products p left join
sell s using (pid)
where s.pid is null;
```

This query searches through the table products for all products that aren't being sold. I did this by implementing a left join where the right side or sell.pid is null. It then prints out these products' names and descriptions.



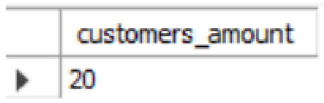
name	description
Super Drive	External CD/DVD/RW
Super Drive	UInternal CD/DVD/RW

3. **How many customers bought SATA drives but not any routers?**



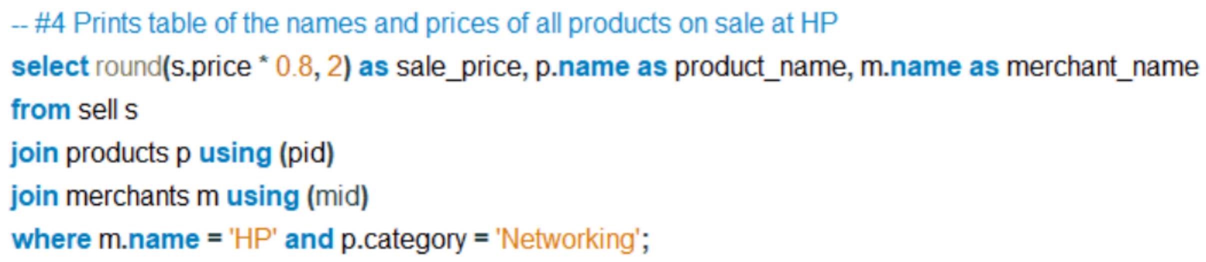
```
-- #3 Prints the count of customers that have purchased sata drives but not routers
select count(*) as customers_amount
from (
    select p.cid
    from place p
    join contain using (oid)
    join products pr using (pid)
    group by p.cid
    having sum(pr.description = '%Router%') = 0 and sum(pr.description like '%Sata%') > 0)  t;
```

This query prints the count of rows returned in a subquery, representing the number of customers who bought a SATA drive but not a router. The subquery groups by every customer and determines if they have purchased these items using a having condition.



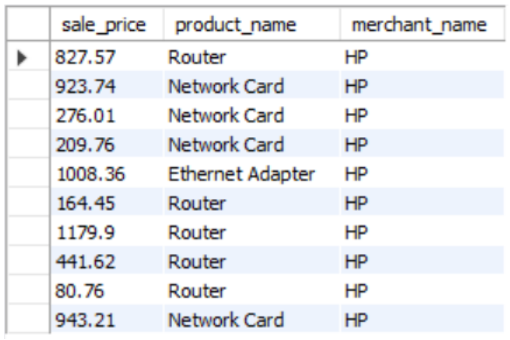
customers_amount
20

4. **HP has a 20% sale on all its Networking products.**



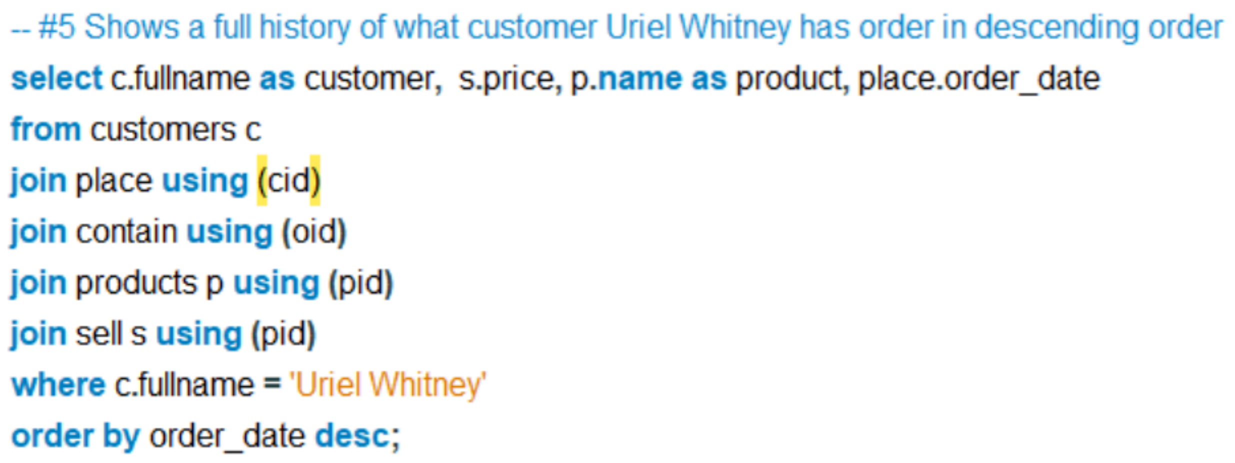
```
-- #4 Prints table of the names and prices of all products on sale at HP
select round(s.price * 0.8, 2) as sale_price, p.name as product_name, m.name as merchant_name
from sell s
join products p using (pid)
join merchants m using (mid)
where m.name = 'HP' and p.category = 'Networking';
```

This query prints a table of all discounted products under the given discount. I did this by printing the product's name, price multiplied by .8 and rounded to two decimal points, and the merchant's name. I also used a where condition to make sure that the discount was only applied to HP products under the Networking category.



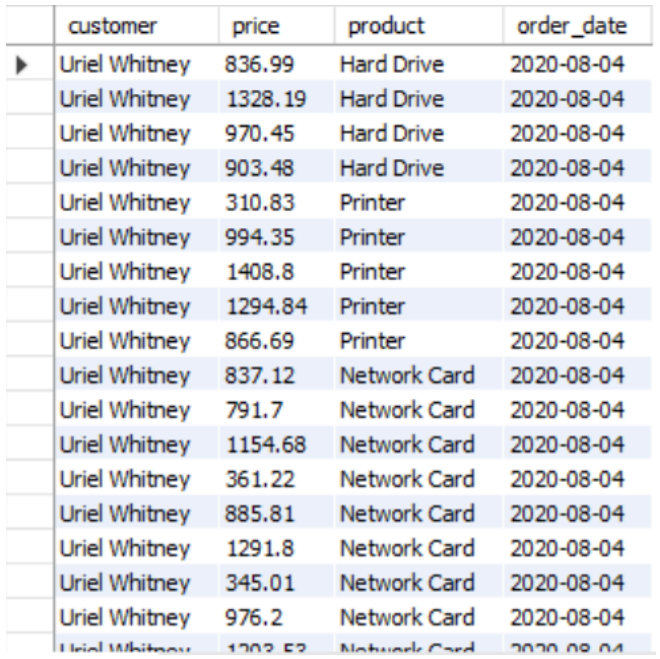
sale_price	product_name	merchant_name
827.57	Router	HP
923.74	Network Card	HP
276.01	Network Card	HP
209.76	Network Card	HP
1008.36	Ethernet Adapter	HP
164.45	Router	HP
1179.9	Router	HP
441.62	Router	HP
80.76	Router	HP
943.21	Network Card	HP

5. **What did Uriel Whitney order?**



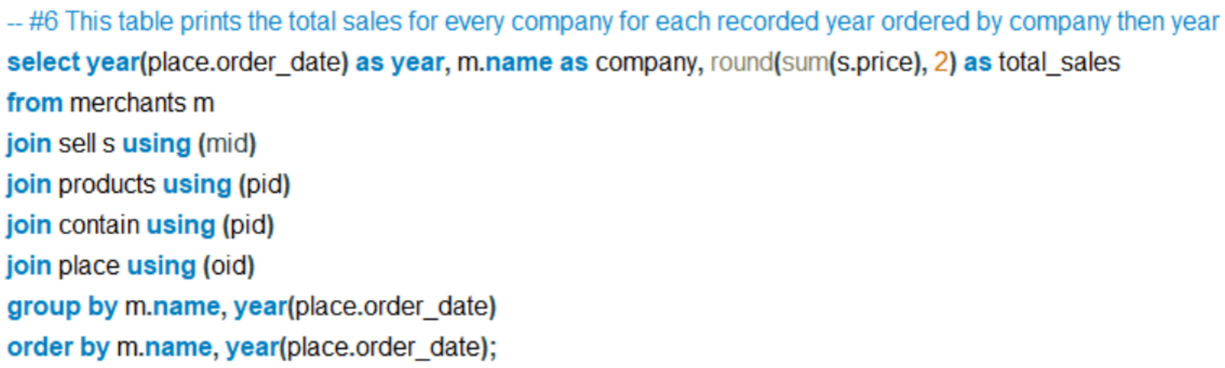
```sql
-- #5 Shows a full history of what customer Uriel Whitney has order in descending order
select c.fullname as customer,  s.price, p.name as product, place.order_date
from customers c
join place using (cid)
join contain using (oid)
join products p using (pid)
join sell s using (pid)
where c.fullname = 'Uriel Whitney'
order by order_date desc;
```

This query prints a table of a specific customer, Uriel Whitney's, entire purchase history, price, product name, and date included. I did this by joining tables customers, place, contain, products, and sell with join using statements. I used the where condition to only show Uriel Whitney's data and ordered it by date descending to get a timeline of his purchases.



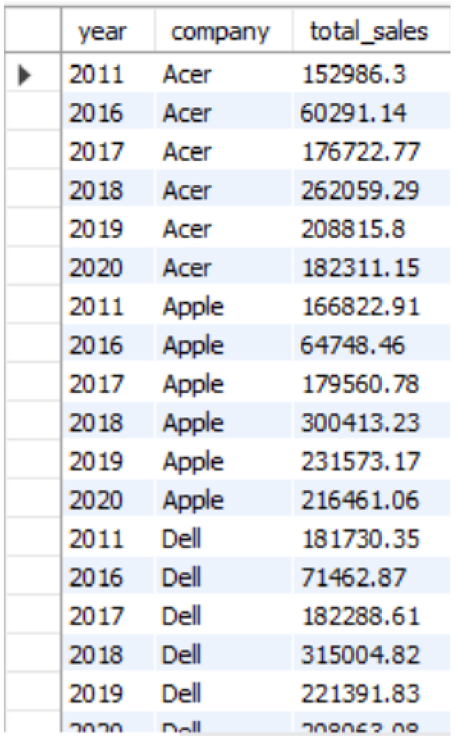
customer	price	product	order_date
Uriel Whitney	836.99	Hard Drive	2020-08-04
Uriel Whitney	1328.19	Hard Drive	2020-08-04
Uriel Whitney	970.45	Hard Drive	2020-08-04
Uriel Whitney	903.48	Hard Drive	2020-08-04
Uriel Whitney	310.83	Printer	2020-08-04
Uriel Whitney	994.35	Printer	2020-08-04
Uriel Whitney	1408.8	Printer	2020-08-04
Uriel Whitney	1294.84	Printer	2020-08-04
Uriel Whitney	866.69	Printer	2020-08-04
Uriel Whitney	837.12	Network Card	2020-08-04
Uriel Whitney	791.7	Network Card	2020-08-04
Uriel Whitney	1154.68	Network Card	2020-08-04
Uriel Whitney	361.22	Network Card	2020-08-04
Uriel Whitney	885.81	Network Card	2020-08-04
Uriel Whitney	1291.8	Network Card	2020-08-04
Uriel Whitney	345.01	Network Card	2020-08-04
Uriel Whitney	976.2	Network Card	2020-08-04
Uriel Whitney	1203.52	Network Card	2020-08-04

6. **List the annual total sales for each company.**



```
-- #6 This table prints the total sales for every company for each recorded year ordered by company then year
select year(place.order_date) as year, m.name as company, round(sum(s.price), 2) as total_sales
from merchants m
join sell s using (mid)
join products using (pid)
join contain using (pid)
join place using (oid)
group by m.name, year(place.order_date)
order by m.name, year(place.order_date);
```

This query prints a table of the total annual revenue each year per company. I did this by first joining tables merchants, sell, products, contain, and place. I then searched this view for the rounded sum of prices grouped by and ordered by merchant name and order date year using the year() function.

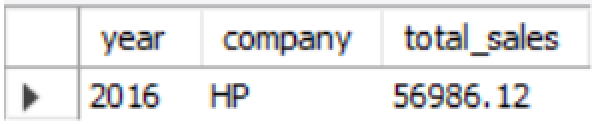


year	company	total_sales
2011	Acer	152986.3
2016	Acer	60291.14
2017	Acer	176722.77
2018	Acer	262059.29
2019	Acer	208815.8
2020	Acer	182311.15
2011	Apple	166822.91
2016	Apple	64748.46
2017	Apple	179560.78
2018	Apple	300413.23
2019	Apple	231573.17
2020	Apple	216461.06
2011	Dell	181730.35
2016	Dell	71462.87
2017	Dell	182288.61
2018	Dell	315004.82
2019	Dell	221391.83
2020	Dell	208063.08

7. **Which company had the highest annual revenue and in what year?**

```sql
-- #7 Prints the one highest annual revenue for any company any year, company name, and the year
select year(place.order_date) as year, m.name as company, round(sum(s.price), 2) as total_sales
from merchants m
join sell s using (mid)
join products using (pid)
join contain using (pid)
join place using (oid)
group by year(place.order_date), m.name
order by total_sales limit 1;
```

This query prints the individual highest performance year by any company all time. I did this by almost exactly repeating my last query, except it is ordered by total_sales, having the largest sum on top, limited by one, so only the highest is shown.

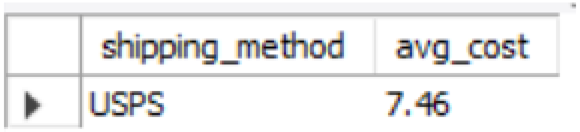


	year	company	total_sales
▶	2016	HP	56986.12

8. **On average, what was the cheapest shipping method used ever?**

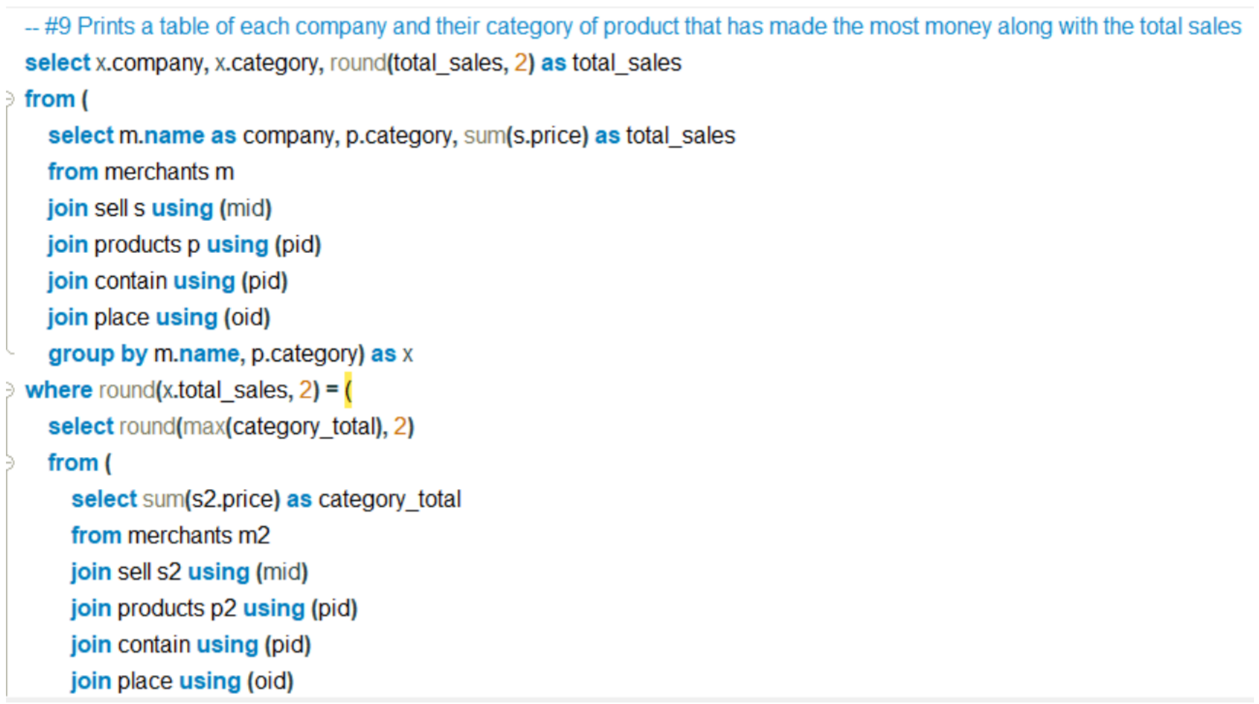
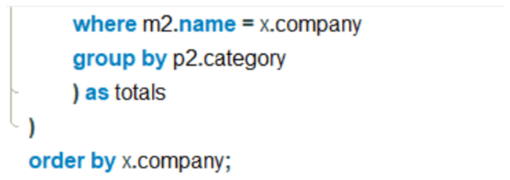
```sql
-- #8 Prints a table of the lowest average shipping method and its average price
select shipping_method, round(avg(shipping_cost), 2) as avg_cost
from orders
group by shipping_method
order by round(avg(shipping_cost), 2) asc
limit 1;
```

This query determines the shipping company that is, on average, the cheapest among all deliveries. I did this by selecting the shipping method and the rounded, average shipping cost, ordering it by price ascending, and limiting it to one, so the lowest value is at the top, and it is the only value.



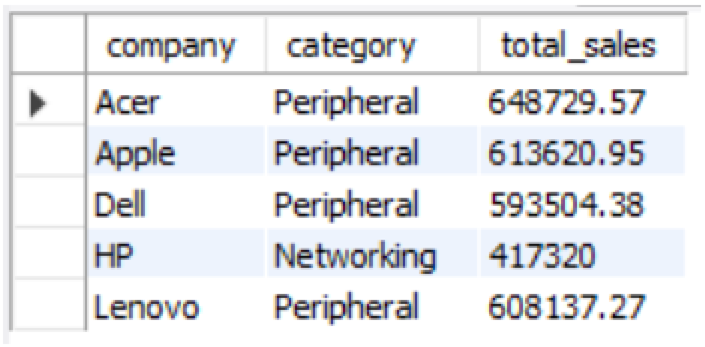
	shipping_method	avg_cost
▶	USPS	7.46

9. **What is the best-selling category for each company?**

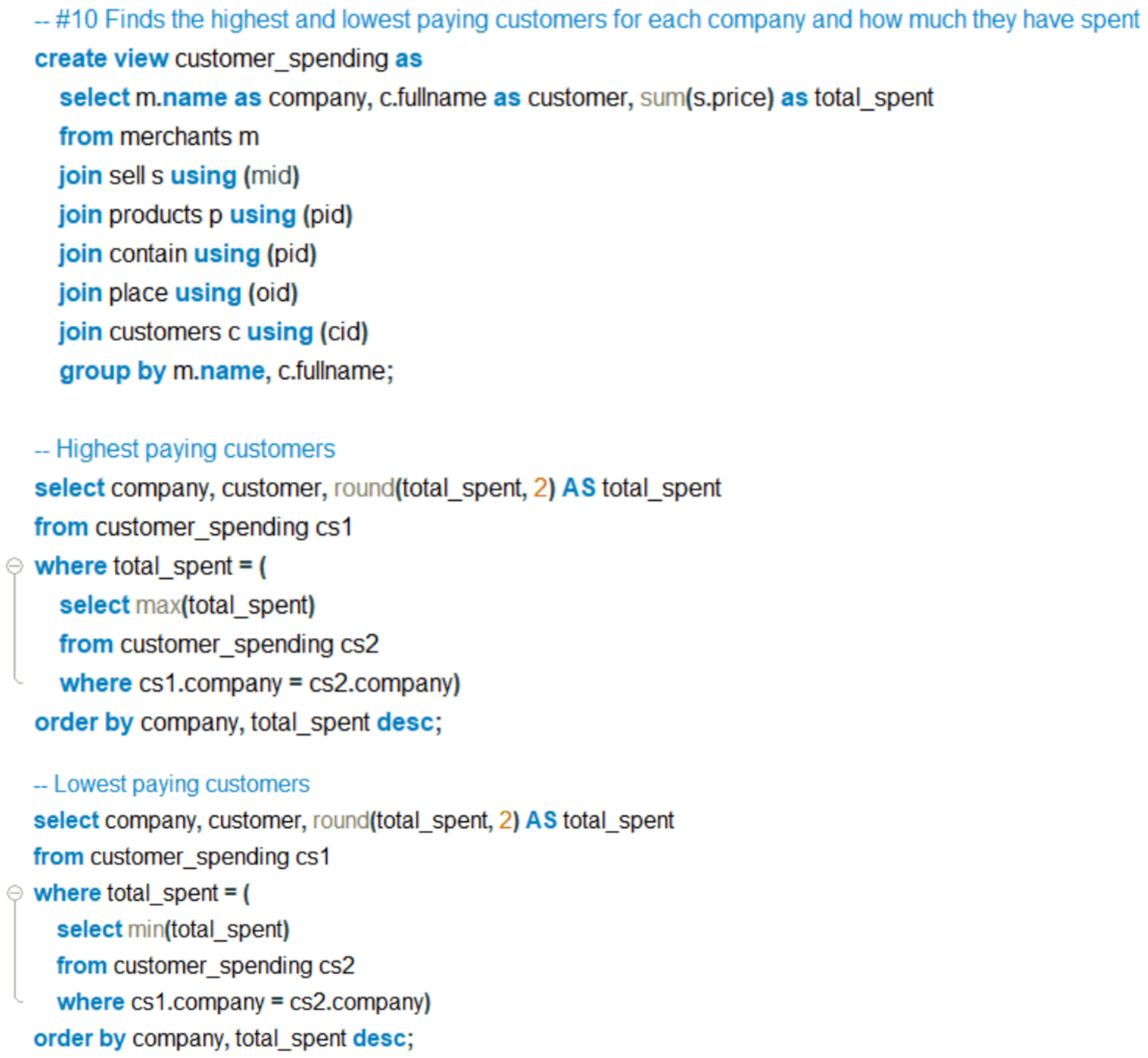
```
-- #9 Prints a table of each company and their category of product that has made the most money along with the total sales
select x.company, x.category, round(total_sales, 2) as total_sales
from (
    select m.name as company, p.category, sum(s.price) as total_sales
    from merchants m
    join sell s using (mid)
    join products p using (pid)
    join contain using (pid)
    join place using (oid)
    group by m.name, p.category) as x
where round(x.total_sales, 2) = (
    select round(max(category_total), 2)
    from (
        select sum(s2.price) as category_total
        from merchants m2
        join sell s2 using (mid)
        join products p2 using (pid)
        join contain using (pid)
        join place using (oid)
        where m2.name = x.company
        group by p2.category
        ) as totals
    )
 order by x.company;
```

This query prints a table displaying each company and its most successful product category. I did this by creating an inner table x which joins merchants, sell, products, contain, and place to find the total sales for each company category group. The subquery totals finds these values as well and then finds the max of category total comparing the totals companies to x companies. The query is then ordered alphabetically by company name.



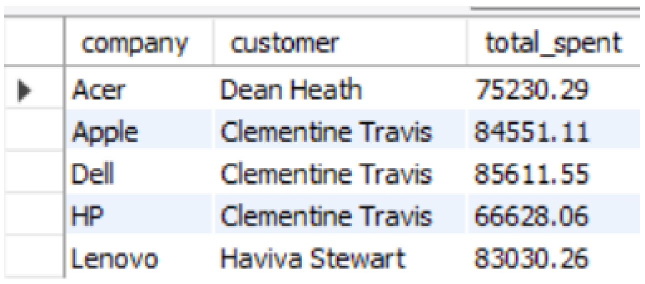
company	category	total_sales
Acer	Peripheral	648729.57
Apple	Peripheral	613620.95
Dell	Peripheral	593504.38
HP	Networking	417320
Lenovo	Peripheral	608137.27

10. For each company, find out which customers have spent the most and the least amounts.

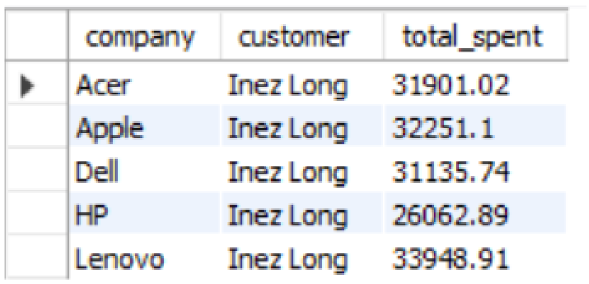


```
-- #10 Finds the highest and lowest paying customers for each company and how much they have spent
create view customer_spending as
  select m.name as company, c.fullname as customer, sum(s.price) as total_spent
  from merchants m
  join sell s using (mid)
  join products p using (pid)
  join contain using (pid)
  join place using (oid)
  join customers c using (cid)
  group by m.name, c.fullname;

-- Highest paying customers
select company, customer, round(total_spent, 2) AS total_spent
from customer_spending cs1
where total_spent = (
  select max(total_spent)
  from customer_spending cs2
  where cs1.company = cs2.company)
order by company, total_spent desc;

-- Lowest paying customers
select company, customer, round(total_spent, 2) AS total_spent
from customer_spending cs1
where total_spent = (
  select min(total_spent)
  from customer_spending cs2
  where cs1.company = cs2.company)
order by company, total_spent desc;
```

These queries work to find the highest and lowest paying customers for each company. First, I started by creating a view of customer spending that joins merchants, sell, products, contain, place, and customers to avoid repetitive coding. Then I found the highest paying customers by printing the company name, customer name, and total spent of price values that equal to the maximum total spent for each respective company, found in the subquery. I then repeated this process but switched key max for min to find the lowest paying customers.



company	customer	total_spent
Acer	Dean Heath	75230.29
Apple	Clementine Travis	84551.11
Dell	Clementine Travis	85611.55
HP	Clementine Travis	66628.06
Lenovo	Haviva Stewart	83030.26



company	customer	total_spent
Acer	Inez Long	31901.02
Apple	Inez Long	32251.1
Dell	Inez Long	31135.74
HP	Inez Long	26062.89
Lenovo	Inez Long	33948.91