Erin Zahner

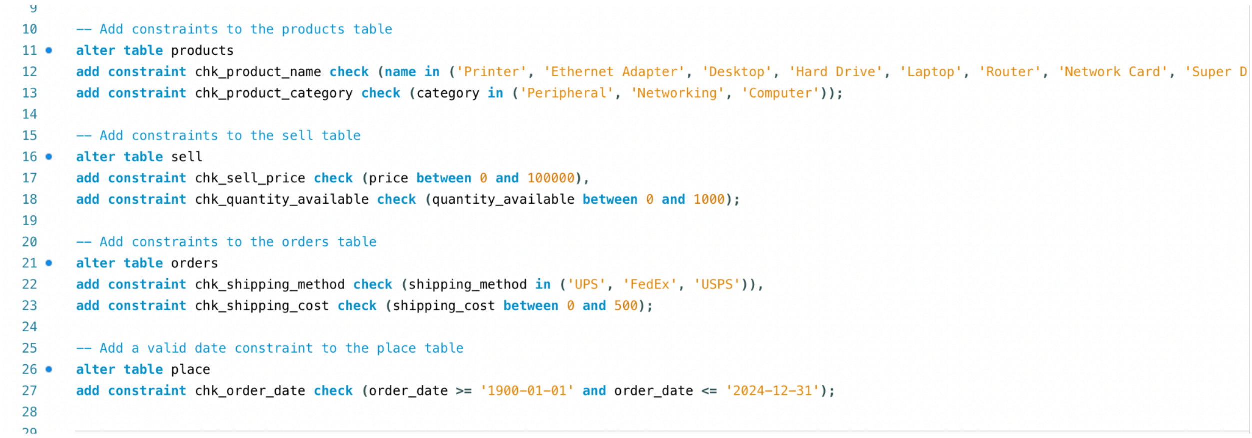
Dr. F

CSC 351

8 October 2024

Assignment 3

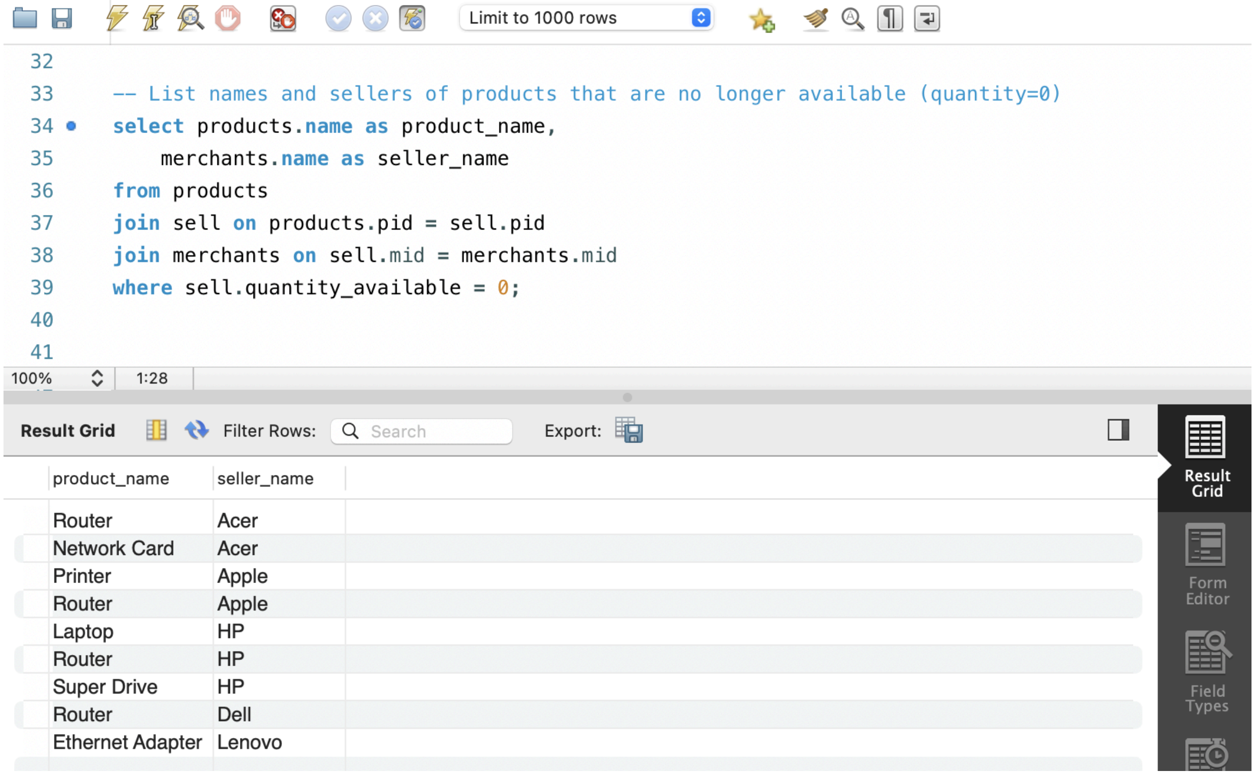
Set-Up



```sql
-- Add constraints to the products table
alter table products
add constraint chk_product_name check (name in ('Printer', 'Ethernet Adapter', 'Desktop', 'Hard Drive', 'Laptop', 'Router', 'Network Card', 'Super D
add constraint chk_product_category check (category in ('Peripheral', 'Networking', 'Computer'));

-- Add constraints to the sell table
alter table sell
add constraint chk_sell_price check (price between 0 and 100000),
add constraint chk_quantity_available check (quantity_available between 0 and 1000);

-- Add constraints to the orders table
alter table orders
add constraint chk_shipping_method check (shipping_method in ('UPS', 'FedEx', 'USPS')),
add constraint chk_shipping_cost check (shipping_cost between 0 and 500);

-- Add a valid date constraint to the place table
alter table place
add constraint chk_order_date check (order_date >= '1900-01-01' and order_date <= '2024-12-31');
```

To set up the database in my workbench, I first imported all the CSV files using ImportWizard. I ensured that each of the files had the proper column names before importing them into my system by first editing the headings. Once everything was in one place, I manually added the primary keys and foreign keys in SQL by checking the necessary boxes for each table. To do this, I right-clicked on my table, selected “alter table”, and was able to select different attributes of the tables. Finally, I ensured all the tables followed the correct constraints in the assignment instructions. For the valid date constraint, I allowed for the dates to be between the beginning of 1900 and the end of this year, 2024.

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

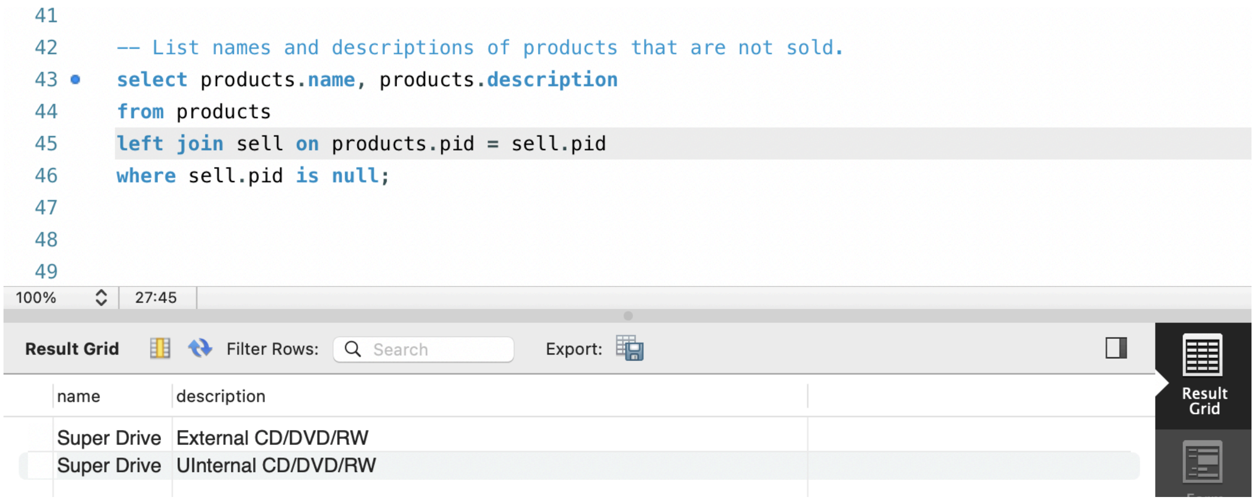


```sql

-- List names and sellers of products that are no longer available (quantity=0)
select products.name as product_name,
    merchants.name as seller_name
from products
join sell on products.pid = sell.pid
join merchants on sell.mid = merchants.mid
where sell.quantity_available = 0;


```

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

In the query above, I began by selecting the product names and the merchant names from the product table and performed joins on the sell and merchant tables. I joined the pid from products on the pid from sells, as well as, the mid from sell with the mid from merchants. Then, to get the desired answer where the quantity = 0, I added the last line which selects all entries in the sell table where the quantity available is 0. Finally, we can see in the resulting table that there were several products from different merchants that were no longer available.

List names and descriptions of products that are not sold.

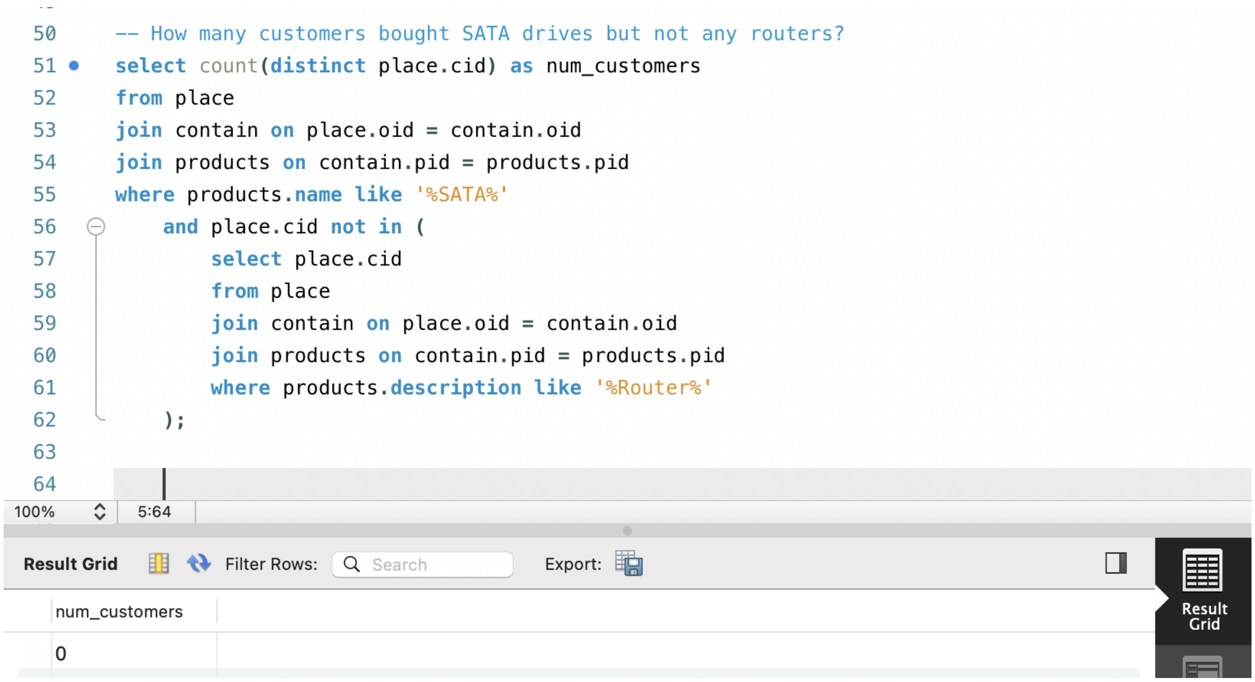


```
-- List names and descriptions of products that are not sold.
select products.name, products.description
from products
left join sell on products.pid = sell.pid
where sell.pid is null;
```

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

In the query above, I started by selecting the product names and descriptions from the products table. Then, I performed a left join between the products table and the sell table on their respective pid columns, linking the product ID from both tables. To find the products that are not sold, I added a condition in the where clause that filters for rows where sell.pid is null, meaning there is no matching entry in the sell table, indicating the product is not currently being sold. As a result, I found that there were two super drives that were not sold.

How many customers bought SATA drives but not any routers?

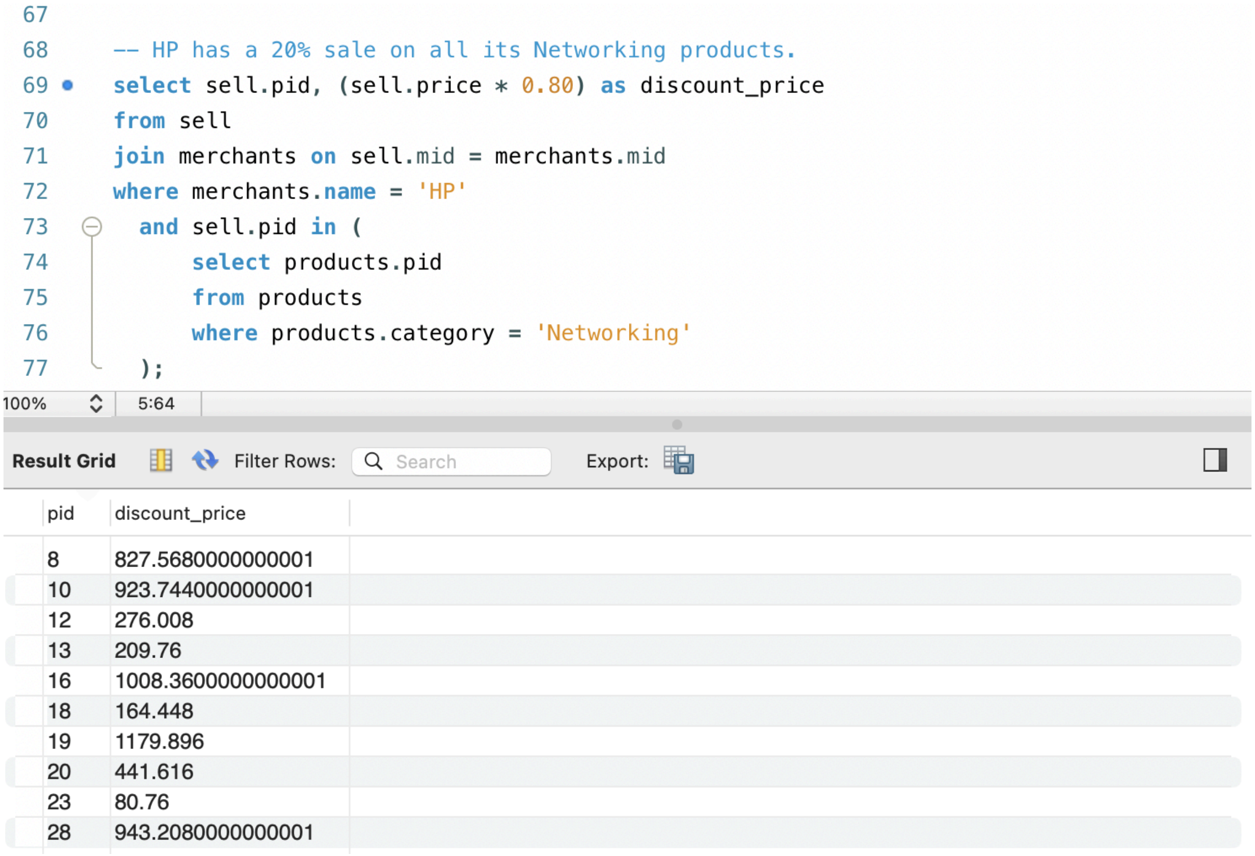


```
-- How many customers bought SATA drives but not any routers?
select count(distinct place.cid) as num_customers
from place
join contain on place.oid = contain.oid
join products on contain.pid = products.pid
where products.name like '%SATA%'
    and place.cid not in (
        select place.cid
        from place
        join contain on place.oid = contain.oid
        join products on contain.pid = products.pid
        where products.description like '%Router%'
    );
```

num_customers
0

In the query above, I started by selecting the distinct customer IDs from the place table and joined it with the contain table on the oid to match orders with products. I also joined the products table on the pid from contain to retrieve product details. Then, I filtered the results to include only products where the name contains "SATA," identifying customers who bought SATA drives. To exclude customers who also bought routers, I added a not in clause. This subquery selects customer IDs from place, contain, and products but filters for products with descriptions containing "router." By excluding these customer IDs, I ensured that only those who bought SATA drives but did not buy routers were counted. From the result, it appears as though all customers who bought SATA drives also bought routers.

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

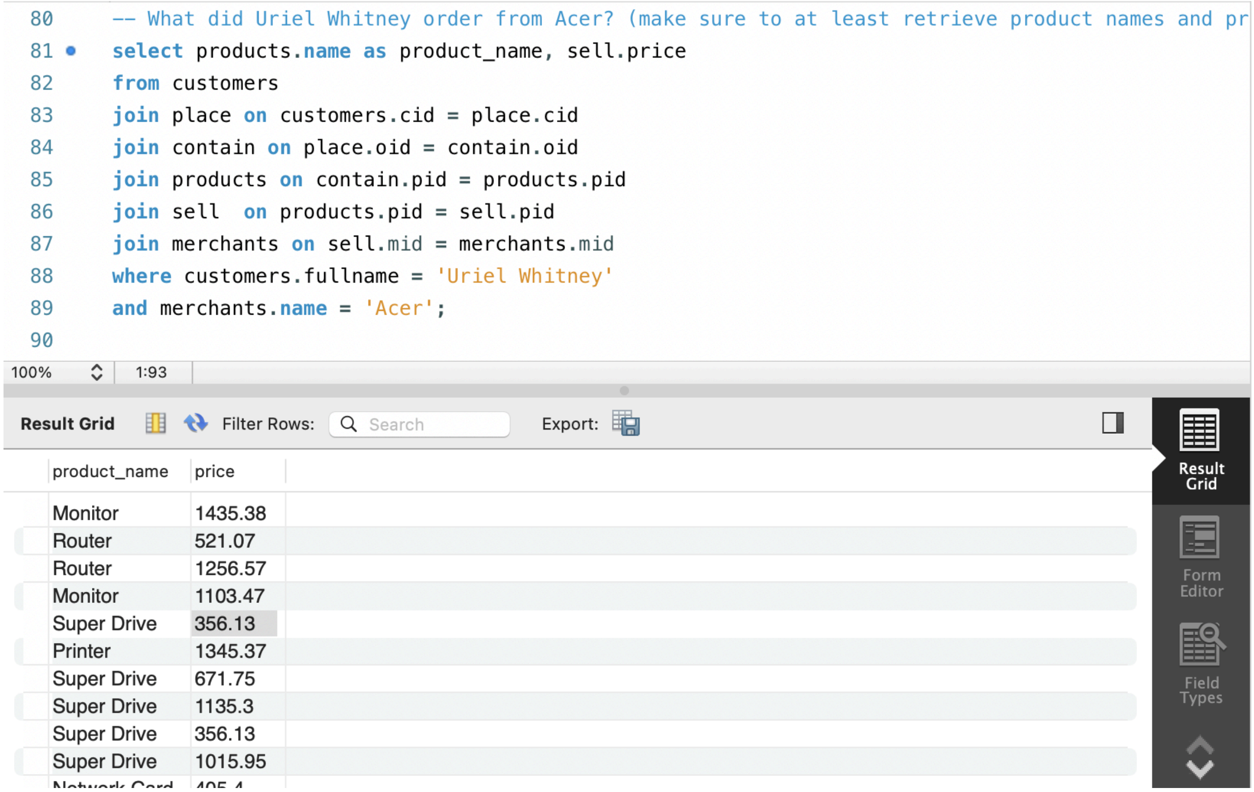


```

-- HP has a 20% sale on all its Networking products.
select sell.pid, (sell.price * 0.80) as discount_price
from sell
join merchants on sell.mid = merchants.mid
where merchants.name = 'HP'
  and sell.pid in (
      select products.pid
      from products
      where products.category = 'Networking'
  );
```

pid	discount_price
8	827.5680000000001
10	923.7440000000001
12	276.008
13	209.76
16	1008.3600000000001
18	164.448
19	1179.896
20	441.616
23	80.76
28	943.2080000000001

In the query above, I began by selecting the product IDs and calculating the discounted price by multiplying the original price by 0.80 to apply the 20% discount. I joined the sell table with the merchants table on the mid to match the sales with the merchant. Then, I filtered the results to include only the merchant named "HP." To ensure the discount is applied only to HP's networking products, I added a subquery. This subquery selects the product IDs from the products table where the category is "Networking." By using these product IDs in the main query, I was able to return only HP's networking products with the discounted prices.

What did Uriel Whitney order from Acer?

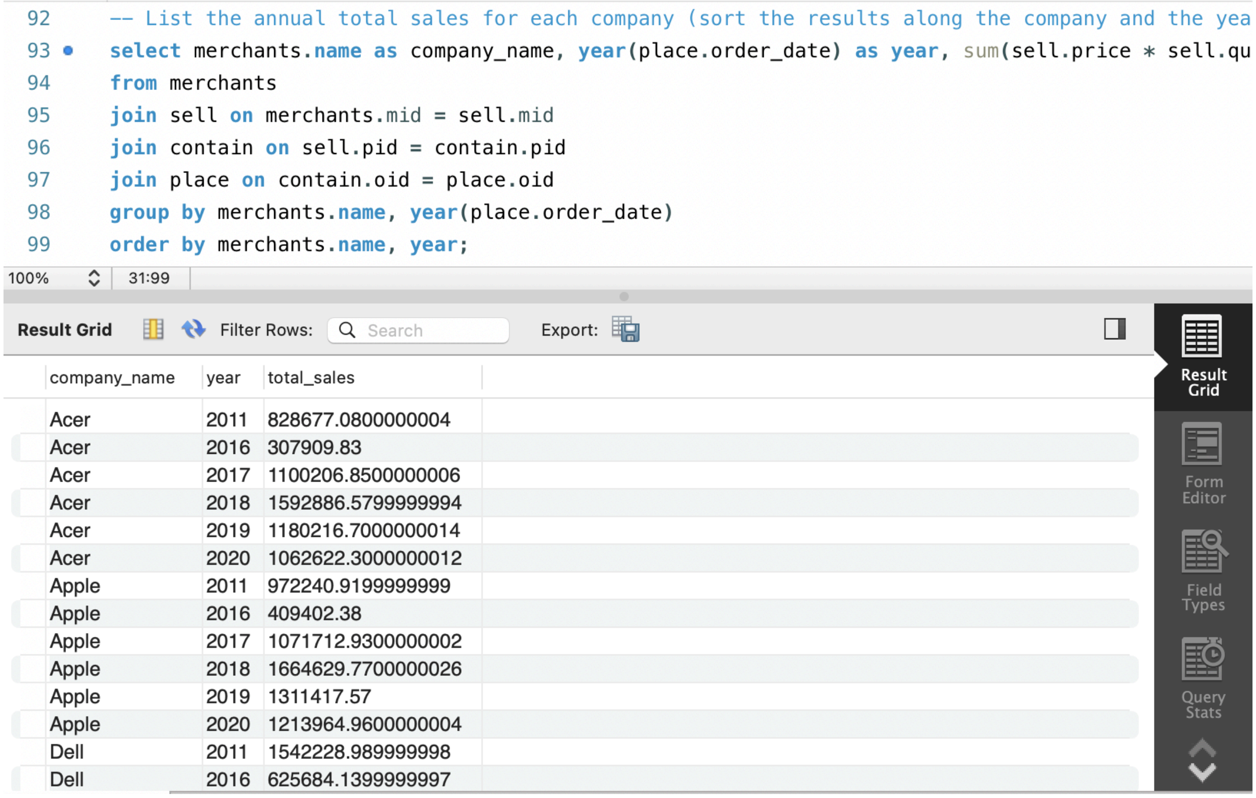


```sql
-- What did Uriel Whitney order from Acer? (make sure to at least retrieve product names and pr
select products.name as product_name, sell.price
from customers
join place on customers.cid = place.cid
join contain on place.oid = contain.oid
join products on contain.pid = products.pid
join sell  on products.pid = sell.pid
join merchants on sell.mid = merchants.mid
where customers.fullname = 'Uriel Whitney'
and merchants.name = 'Acer';
```

product_name	price
Monitor	1435.38
Router	521.07
Router	1256.57
Monitor	1103.47
Super Drive	356.13
Printer	1345.37
Super Drive	671.75
Super Drive	1135.3
Super Drive	356.13
Super Drive	1015.95
Network Card	405.4

In the query above, I started by selecting the product names and prices. I joined the customers table with the place table on the cid to connect customers with their orders. Next, I joined the place table with the contain table using the oid to link orders with the products they contain. I then joined the products table on pid to retrieve product details. To get the prices for the products, I joined the sell table on the pid and linked the sell table with the merchants table on mid to filter for products sold by Acer. Finally, I added the where clause to filter for the customer "Uriel Whitney" and the merchant "Acer," returning the products ordered by Uriel Whitney from Acer along with their prices.

List the annual total sales for each company



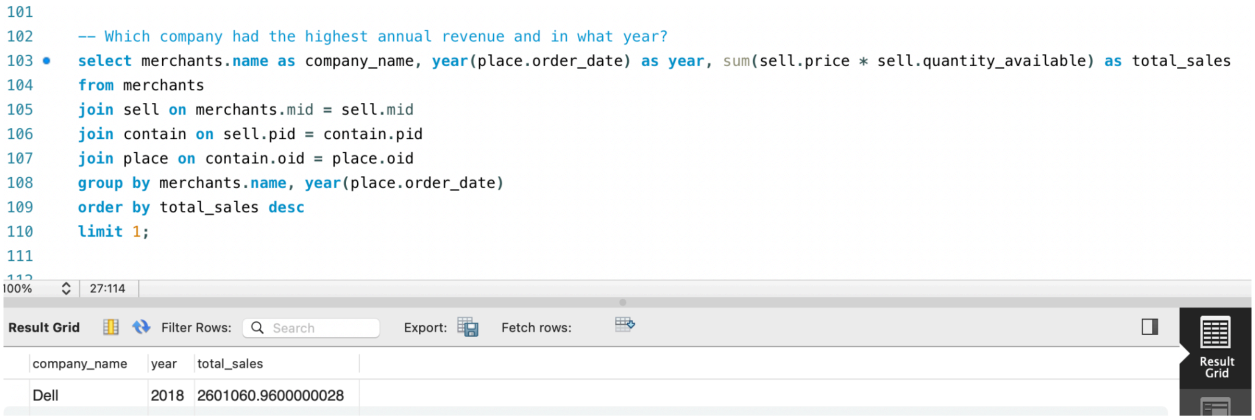
```
-- List the annual total sales for each company (sort the results along the company and the yea
select merchants.name as company_name, year(place.order_date) as year, sum(sell.price * sell.qu
from merchants
join sell on merchants.mid = sell.mid
join contain on sell.pid = contain.pid
join place on contain.oid = place.oid
group by merchants.name, year(place.order_date)
order by merchants.name, year;
```

company_name	year	total_sales
Acer	2011	828677.0800000004
Acer	2016	307909.83
Acer	2017	1100206.8500000006
Acer	2018	1592886.5799999994
Acer	2019	1180216.7000000014
Acer	2020	1062622.3000000012
Apple	2011	972240.9199999999
Apple	2016	409402.38
Apple	2017	1071712.9300000002
Apple	2018	1664629.7700000026
Apple	2019	1311417.57
Apple	2020	1213964.9600000004
Dell	2011	1542228.989999998
Dell	2016	625684.1399999997

In the query above, I began by selecting the company names and the year of the order. I joined the merchants table with the sell table on mid to link each merchant with the products they sell. Then, I joined the sell table with the contain table on pid to connect the products with their corresponding orders. Finally, I joined the place table on oid to access the order dates.

To calculate the total sales for each company, I multiplied the product price by the quantity available and summed these values. I am assuming for the annual sales, each company sells out of all the products they have available. I grouped the results by both the company name and the year of the order to calculate the annual total sales. Finally, I sorted the results first by company name and then by year to display the total sales for each company in chronological order.

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

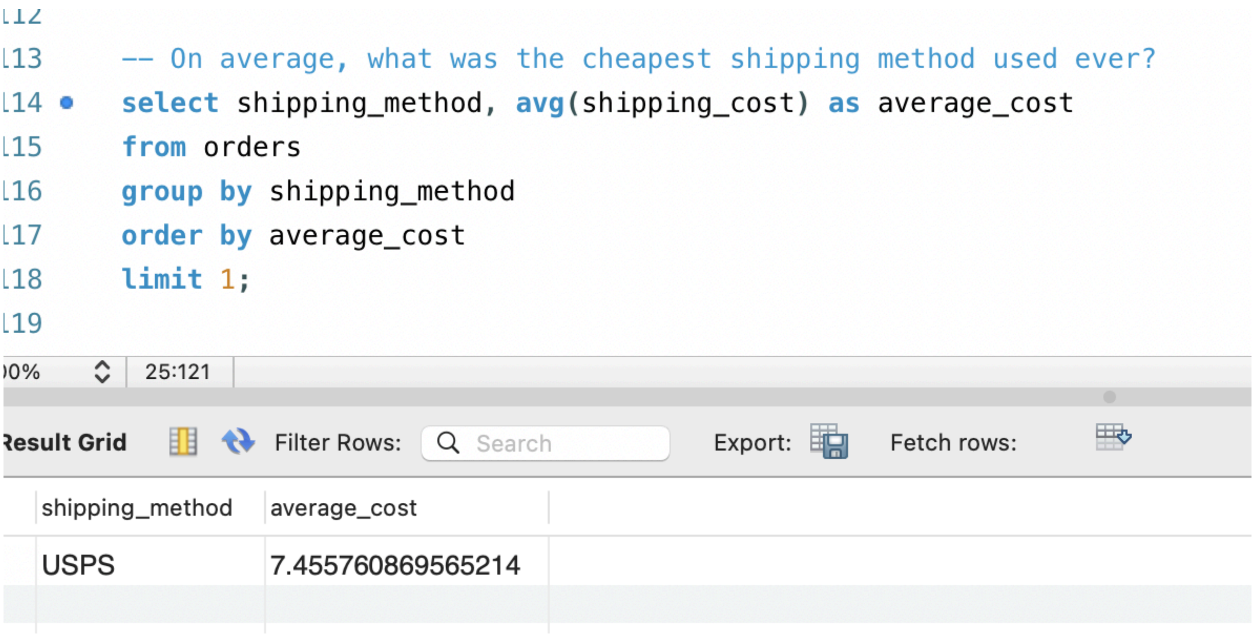


```sql

-- Which company had the highest annual revenue and in what year?
select merchants.name as company_name, year(place.order_date) as year, sum(sell.price * sell.quantity_available) as total_sales
from merchants
join sell on merchants.mid = sell.mid
join contain on sell.pid = contain.pid
join place on contain.oid = place.oid
group by merchants.name, year(place.order_date)
order by total_sales desc
limit 1;

```

company_name	year	total_sales
Dell	2018	2601060.9600000028

In the query above, I began by selecting the company names and the year of the order. I joined the merchants table with the sell table on mid to connect each merchant with the products they sell. Next, I joined the sell table with the contain table using the pid to link products to their corresponding orders. I then joined the place table on oid to access the order dates. To calculate the total annual revenue, I summed the product prices multiplied by the quantity available from the sell table. I grouped the results by company name and year to determine the annual revenue for each company. Finally, I sorted the results in descending order by total sales and limited the output to one entry to find the company with the highest annual revenue and the corresponding year. As a result, we can see that Dell had the highest annual revenue in 2018, bringing in 2.6 million dollars.

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

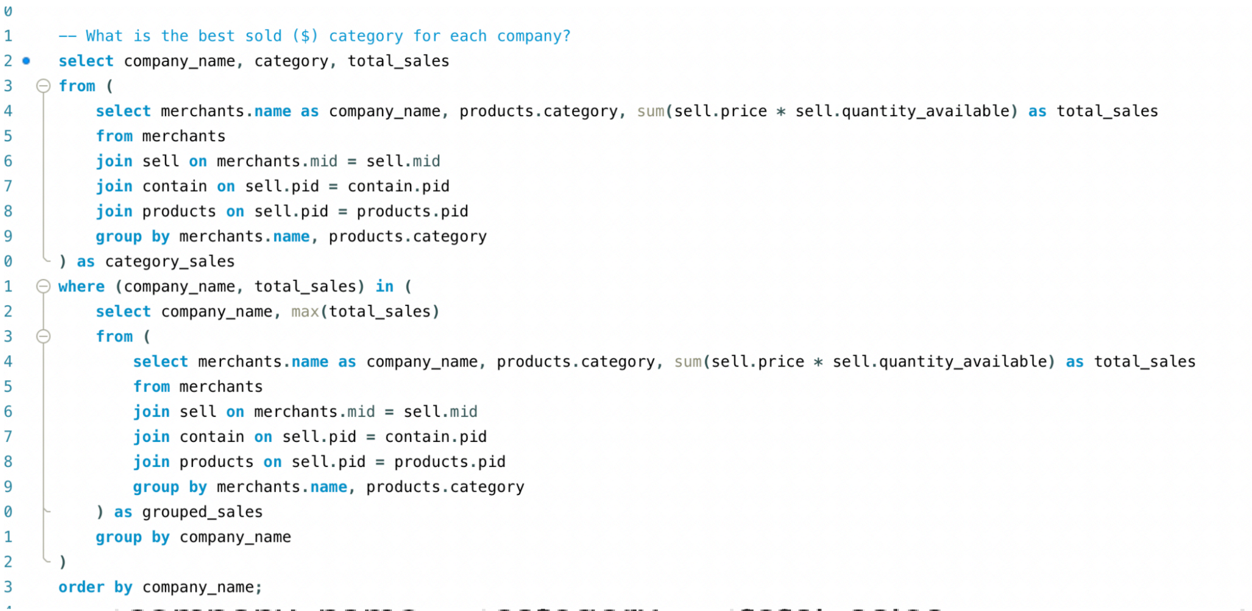


```
-- On average, what was the cheapest shipping method used ever?
select shipping_method, avg(shipping_cost) as average_cost
from orders
group by shipping_method
order by average_cost
limit 1;
```

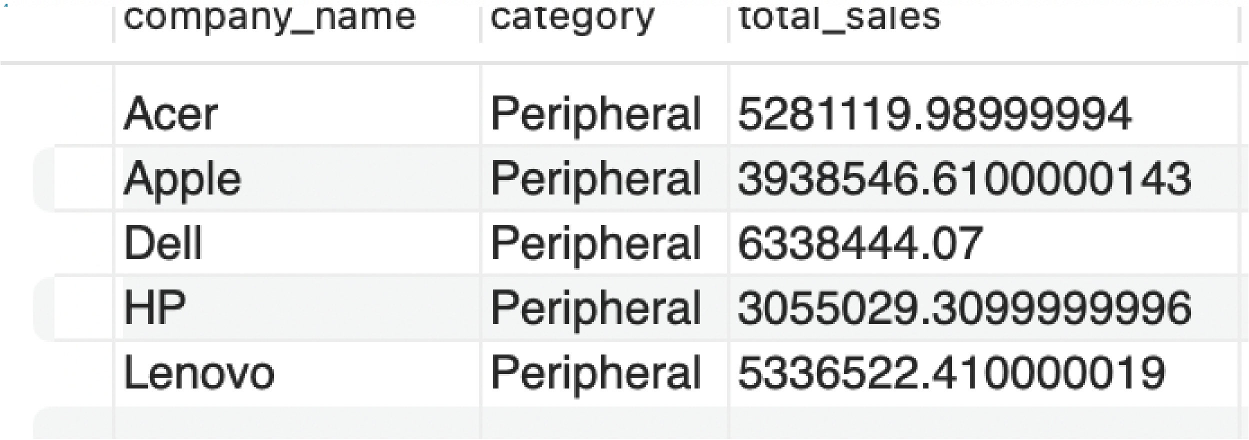
shipping_method	average_cost
USPS	7.455760869565214

In the query above, I started by selecting the shipping methods and calculating the average shipping cost from the orders table. I used the avg function to compute the average cost for each shipping method. To group the results, I applied the group by clause on the shipping_method to ensure that the average cost is calculated for each distinct method. Finally, I sorted the results in ascending order by average cost and limited the output to one entry to identify the cheapest shipping method used on average. As a result, we can see that USPS has the lowest average shipping cost of around $7.45.

What is the best sold ($) category for each company?



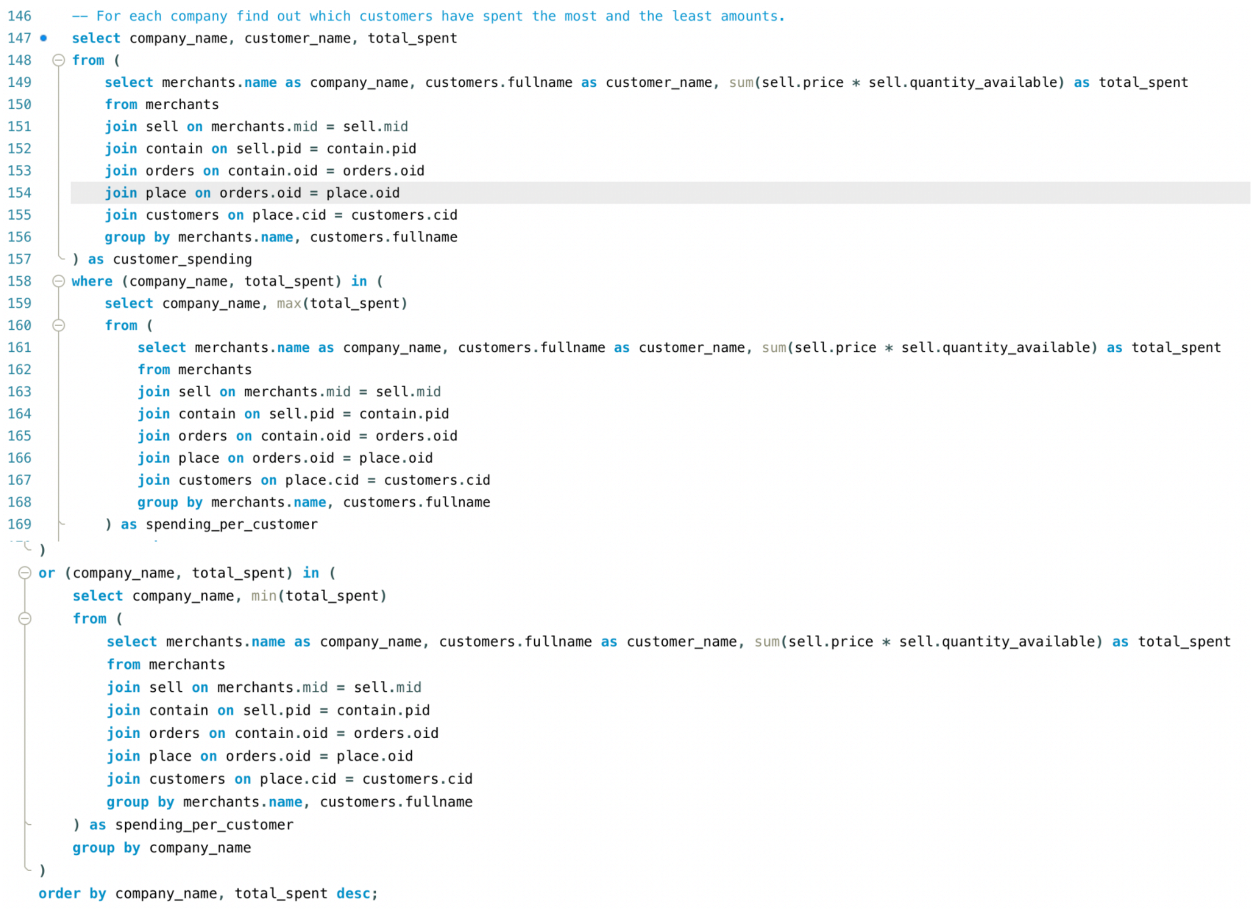
```
-- What is the best sold ($) category for each company?
select company_name, category, total_sales
from (
    select merchants.name as company_name, products.category, sum(sell.price * sell.quantity_available) as total_sales
    from merchants
    join sell on merchants.mid = sell.mid
    join contain on sell.pid = contain.pid
    join products on sell.pid = products.pid
    group by merchants.name, products.category
) as category_sales
where (company_name, total_sales) in (
    select company_name, max(total_sales)
    from (
        select merchants.name as company_name, products.category, sum(sell.price * sell.quantity_available) as total_sales
        from merchants
        join sell on merchants.mid = sell.mid
        join contain on sell.pid = contain.pid
        join products on sell.pid = products.pid
        group by merchants.name, products.category
    ) as grouped_sales
    group by company_name
)
order by company_name;
```



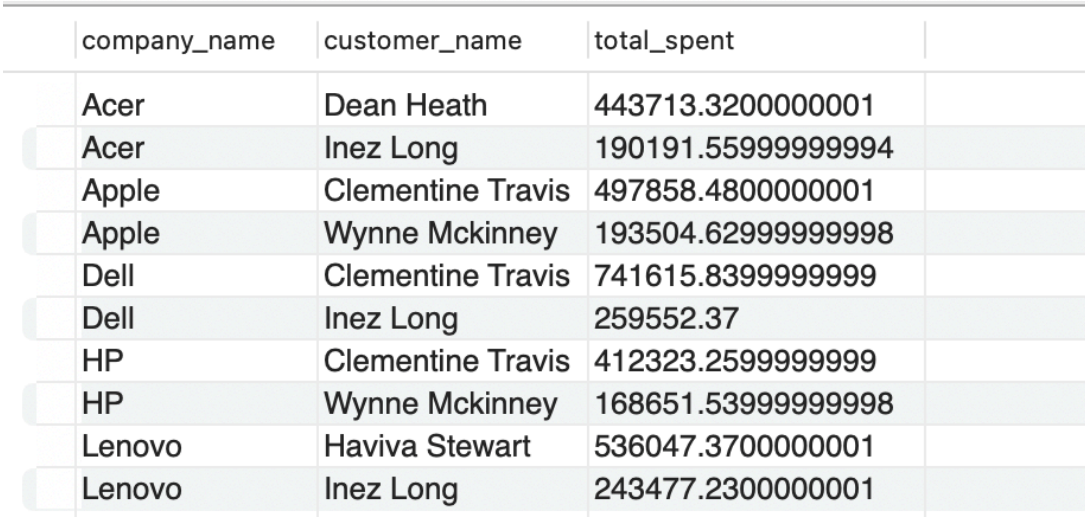
company_name	category	total_sales
Acer	Peripheral	5281119.98999994
Apple	Peripheral	3938546.6100000143
Dell	Peripheral	6338444.07
HP	Peripheral	3055029.3099999996
Lenovo	Peripheral	5336522.410000019

To do this, I joined several tables: merchants, sell, contain, and products. I linked them using the mid from the merchants and sell tables and the pid from the sell and products tables. Next, I grouped the data by company name and product category, calculating total sales by multiplying the price of each product by the quantity available and summing it up for each category. To find the top-selling category for each company, I used a subquery. This subquery calculated the total sales per category for each company and then identified the category with the highest total sales. I filtered the main query to only include these top-selling categories and finally ordered the results by company name to make it easier to read. From this, I got the result that the best selling category for each company was Peripheral.

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



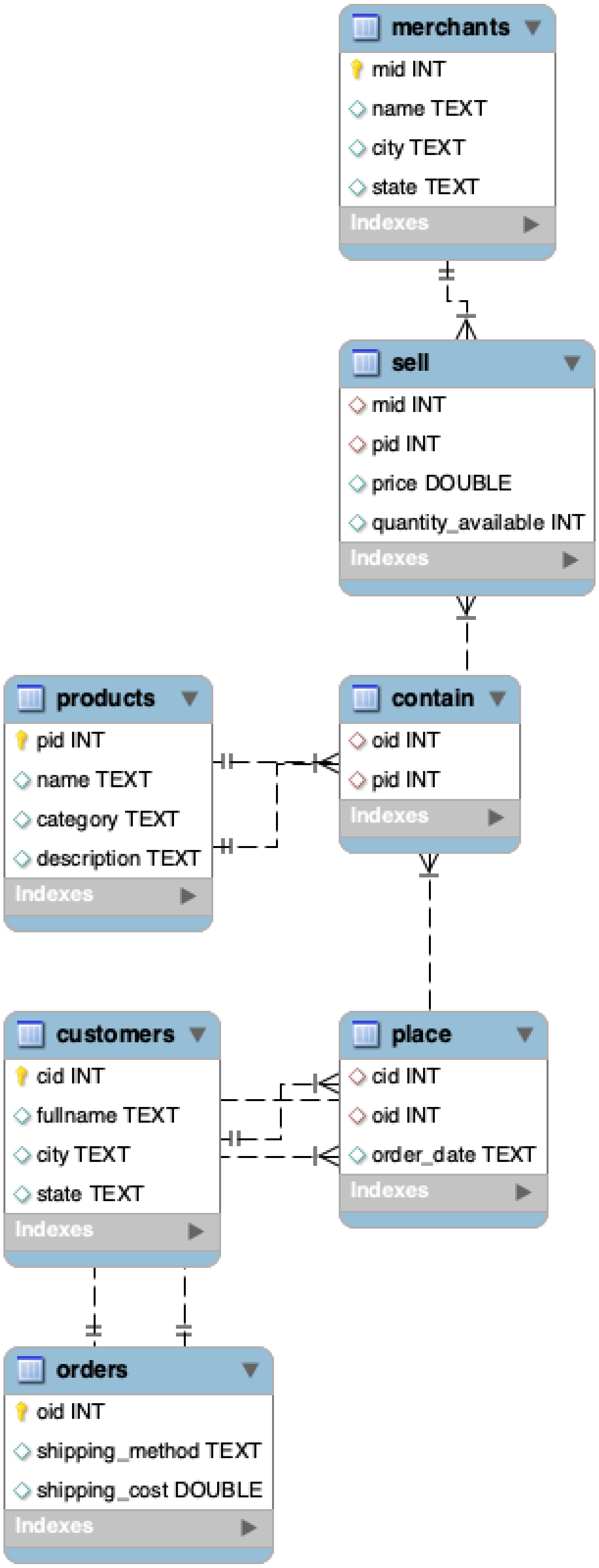
```sql
-- For each company find out which customers have spent the most and the least amounts.
select company_name, customer_name, total_spent
from (
    select merchants.name as company_name, customers.fullname as customer_name, sum(sell.price * sell.quantity_available) as total_spent
    from merchants
    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
    join customers on place.cid = customers.cid
    group by merchants.name, customers.fullname
) as customer_spending
where (company_name, total_spent) in (
    select company_name, max(total_spent)
    from (
        select merchants.name as company_name, customers.fullname as customer_name, sum(sell.price * sell.quantity_available) as total_spent
        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
        join customers on place.cid = customers.cid
        group by merchants.name, customers.fullname
    ) as spending_per_customer
)
or (company_name, total_spent) in (
    select company_name, min(total_spent)
    from (
        select merchants.name as company_name, customers.fullname as customer_name, sum(sell.price * sell.quantity_available) as total_spent
        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
        join customers on place.cid = customers.cid
        group by merchants.name, customers.fullname
    ) as spending_per_customer
    group by company_name
)
order by company_name, total_spent desc;
```



company_name	customer_name	total_spent
Acer	Dean Heath	443713.3200000001
Acer	Inez Long	190191.55999999994
Apple	Clementine Travis	497858.4800000001
Apple	Wynne Mckinney	193504.62999999998
Dell	Clementine Travis	741615.8399999999
Dell	Inez Long	259552.37
HP	Clementine Travis	412323.2599999999
HP	Wynne Mckinney	168651.53999999998
Lenovo	Haviva Stewart	536047.3700000001
Lenovo	Inez Long	243477.2300000001

In the query above, I began by selecting the company name, customer name, and total amount spent from a derived table. In this derived table, I joined the merchants, sell, contain, orders, place, and customers tables to connect each customer with their purchases at different companies. I calculated the total spent by each customer for each company by summing the product prices multiplied by the quantity available from the sell table and grouped the results by both the merchant name and customer name. To identify the customers who spent the most and least at each company, I used two subqueries. The first subquery finds the maximum total spent for each company, while the second subquery finds the minimum total spent. Each subquery groups by company name and retrieves the relevant spending amounts. I filtered the outer query results to include only those customers whose spending matches either the maximum or minimum for their respective companies. Finally, I sorted the output by company name and total spent in descending order to clearly show the customers with the highest and lowest spending for each company.

ER Diagram


merchants
mid INT
name TEXT
city TEXT
state TEXT
Indexes
sell
mid INT
pid INT
price DOUBLE
quantity_available INT
Indexes
products
pid INT
name TEXT
category TEXT
description TEXT
Indexes
contain
oid INT
pid INT
Indexes
customers
cid INT
fullname TEXT
city TEXT
state TEXT
Indexes
place
cid INT
oid INT
order_date TEXT
Indexes
orders
oid INT
shipping_method TEXT
shipping_cost DOUBLE
Indexes