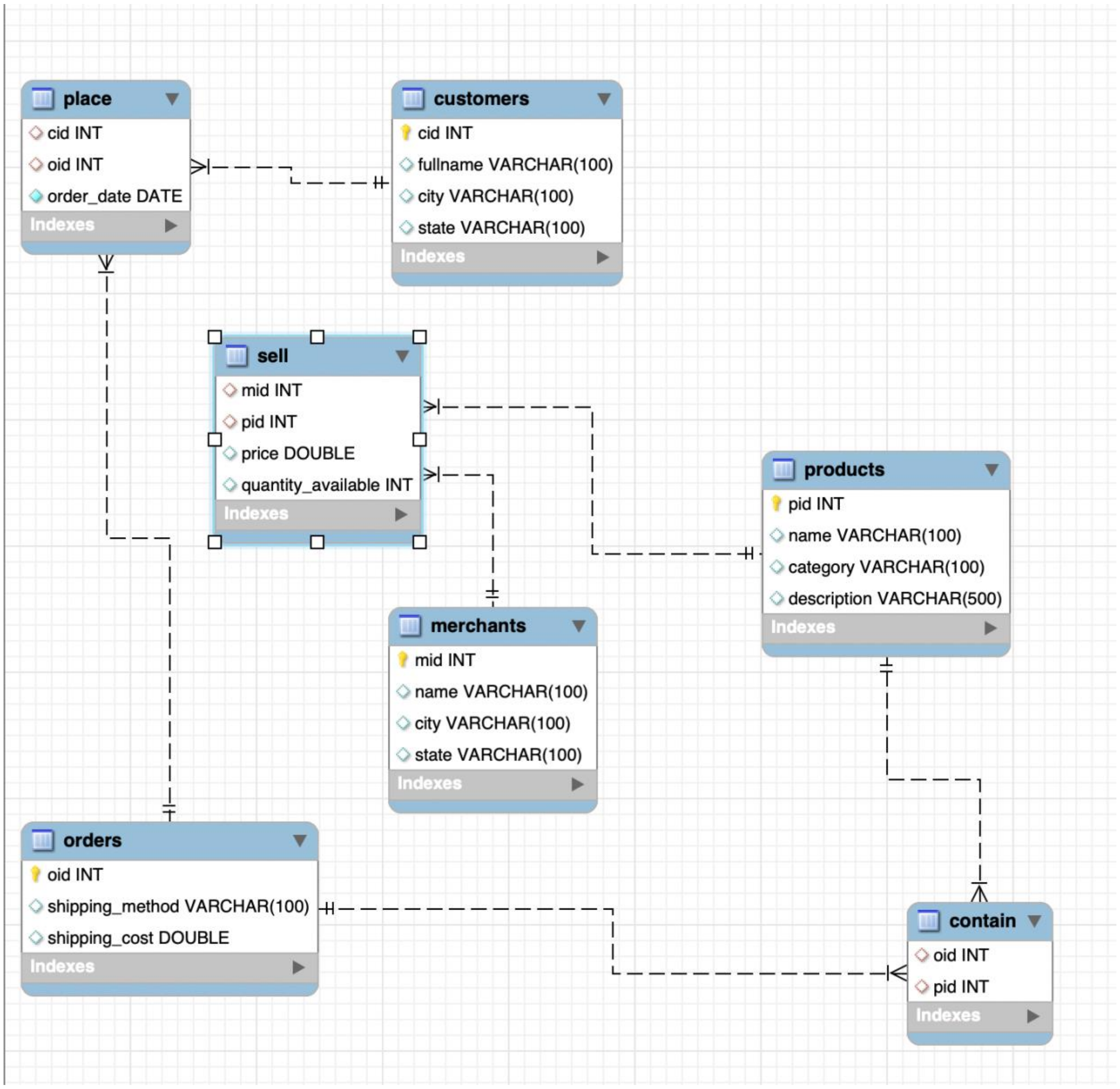
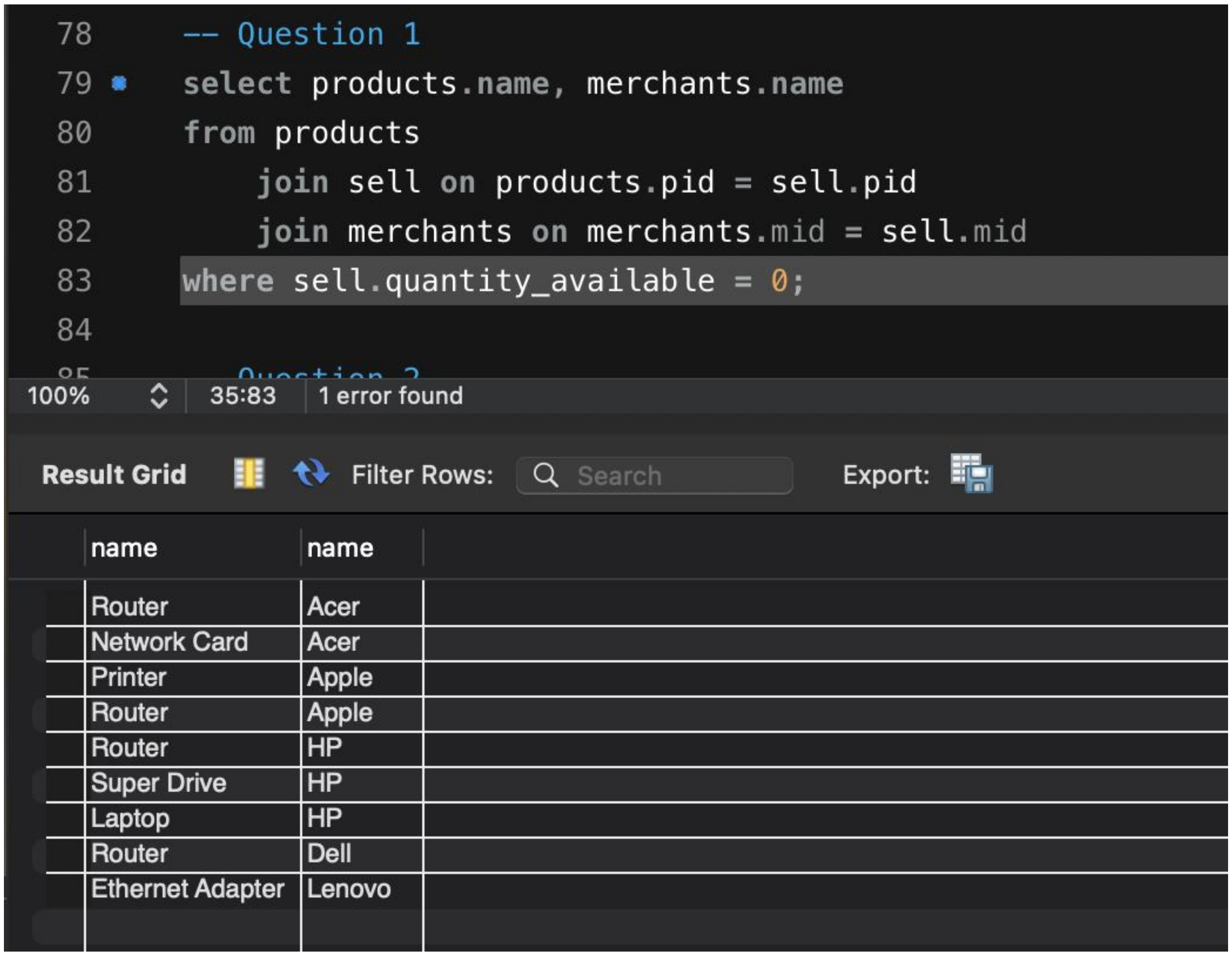
Title: DB Assignment 3

Name: Devon Reing

Date: 8 October 2024



The ER diagram for this picture illustrated above shows the relationships between the 7 tables and the attributes for each entity set. The tables sell, contain, and place are all relationships that connect the remaining entity sets which can be seen through the foreign keys. Each of the foreign keys comes from the connected entity's primary key.

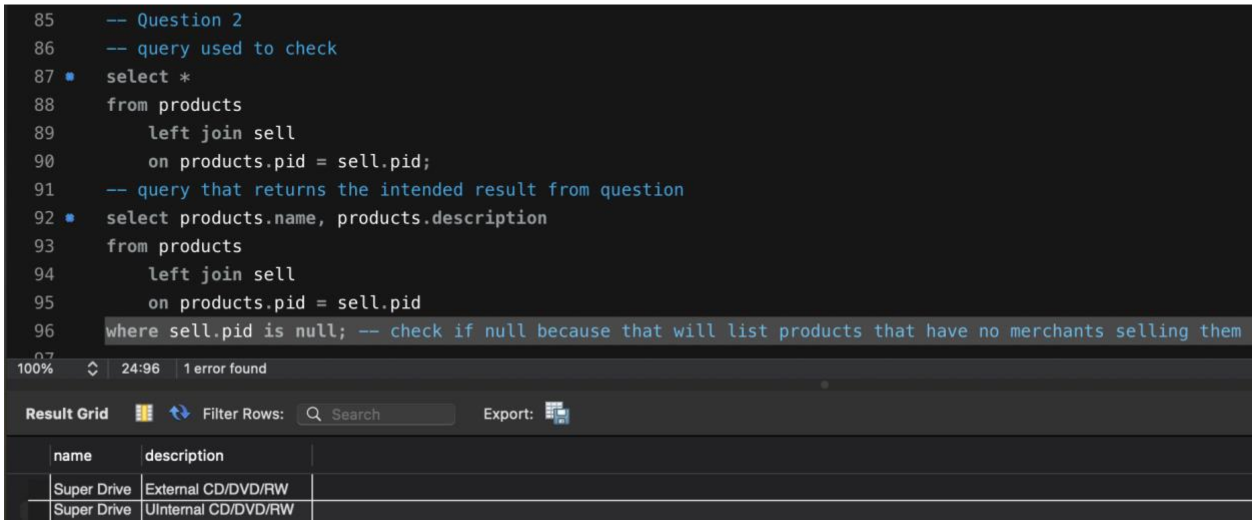


```
-- Question 1
select products.name, merchants.name
from products
    join sell on products.pid = sell.pid
    join merchants on merchants.mid = sell.mid
where sell.quantity_available = 0;
```

```
100%    35:83    1 error found
```

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

The first query aims to list every product that is no longer available. This is done through checking when the quantity available is 0 in the sell table for each product. To get each product along with who sells it an inner join is done on products, sell, and merchants through the id fields for each respective table. The attributes shown in the output is provided by the select statement, which provides both the merchant name and the product name.

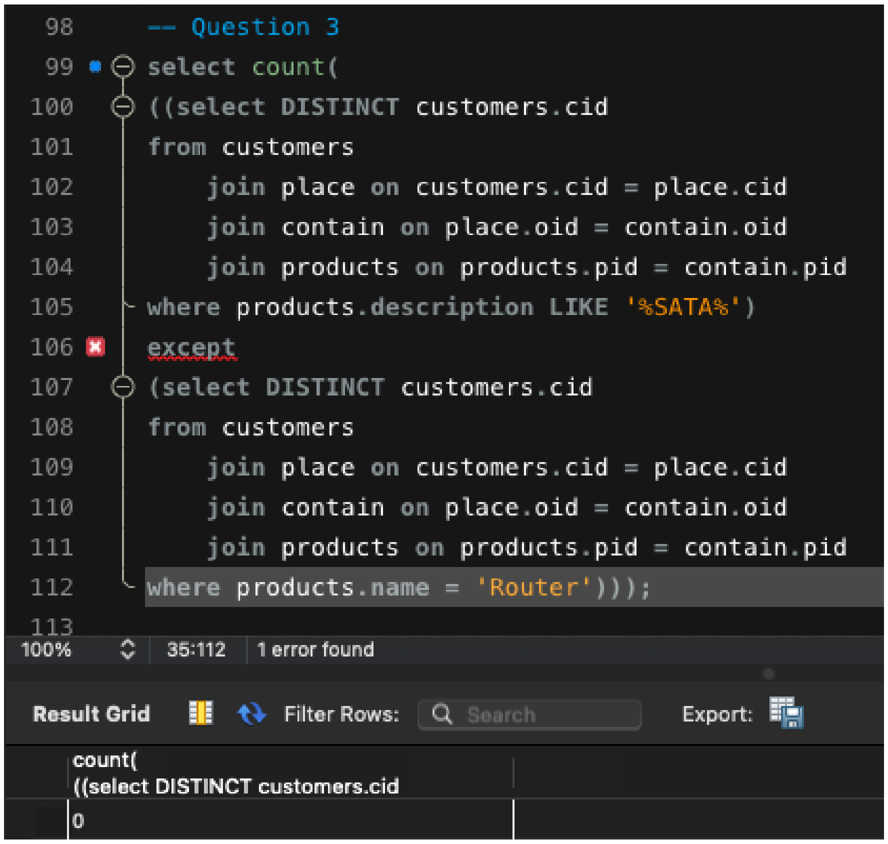


```
-- Question 2
-- query used to check
select *
from products
    left join sell
    on products.pid = sell.pid;
-- query that returns the intended result from question
select products.name, products.description
from products
    left join sell
    on products.pid = sell.pid
where sell.pid is null; -- check if null because that will list products that have no merchants selling them
```

```
100%    24:96   1 error found
```

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

The second query aims to list the products in the products table that are not sold by any merchant. To achieve this, a left join is done on products with sell using the product id attributes in each table. With a left join, every entry from the products (left side) will be kept in the new combined table. If an entry with the same product id is not present in the sell table, the remaining columns from the sell table will be filled with null values. Using the where statement, the products to appear in the output are filtered by checking for a null value in the sell column for product id. To format the output and avoid all columns from appearing in the output table, the select statement limits the columns that appear in the output table to the products name and description.

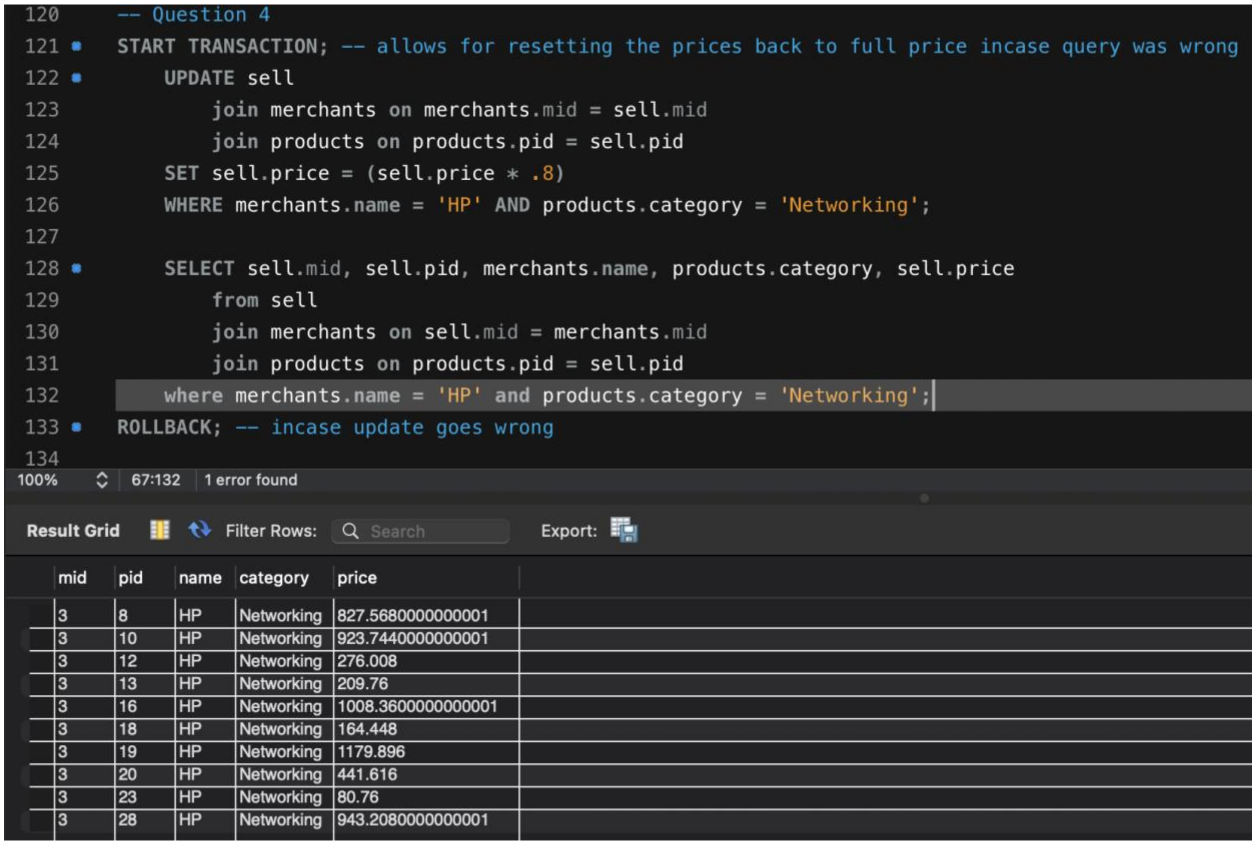


```
-- Question 3
select count(
((select DISTINCT customers.cid
from customers
    join place on customers.cid = place.cid
    join contain on place.oid = contain.oid
    join products on products.pid = contain.pid
where products.description LIKE '%SATA%')
except
(select DISTINCT customers.cid
from customers
    join place on customers.cid = place.cid
    join contain on place.oid = contain.oid
    join products on products.pid = contain.pid
where products.name = 'Router')));
```

```
100%   35:112   1 error found
```

```
count(
((select DISTINCT customers.cid
0
```

The third query aims to find the number of customers that bought SATA drive products but not any routers. To achieve this, two separate subqueries are executed and joined together through an except statement to only show the values that appear in the first subquery and not the second. The first subquery selects the distinct customers that have purchased a drive with SATA in the description. To make this possible, customers, place, contain, and products are combined using an inner join on the id values present in the tables. The distinct keyword ensures a customer is not included twice in the list if they have bought multiple products that satisfy the criteria. The second subquery follows a similar process and combines the same tables in the same way. Then each distinct customer id that has purchased a product with the name router is selected. The except statement removes any values returned in the second subquery from the results returned from the first subquery. Finally the main query takes the returned customer ids and counts the number of them that are present to return that in the output table.



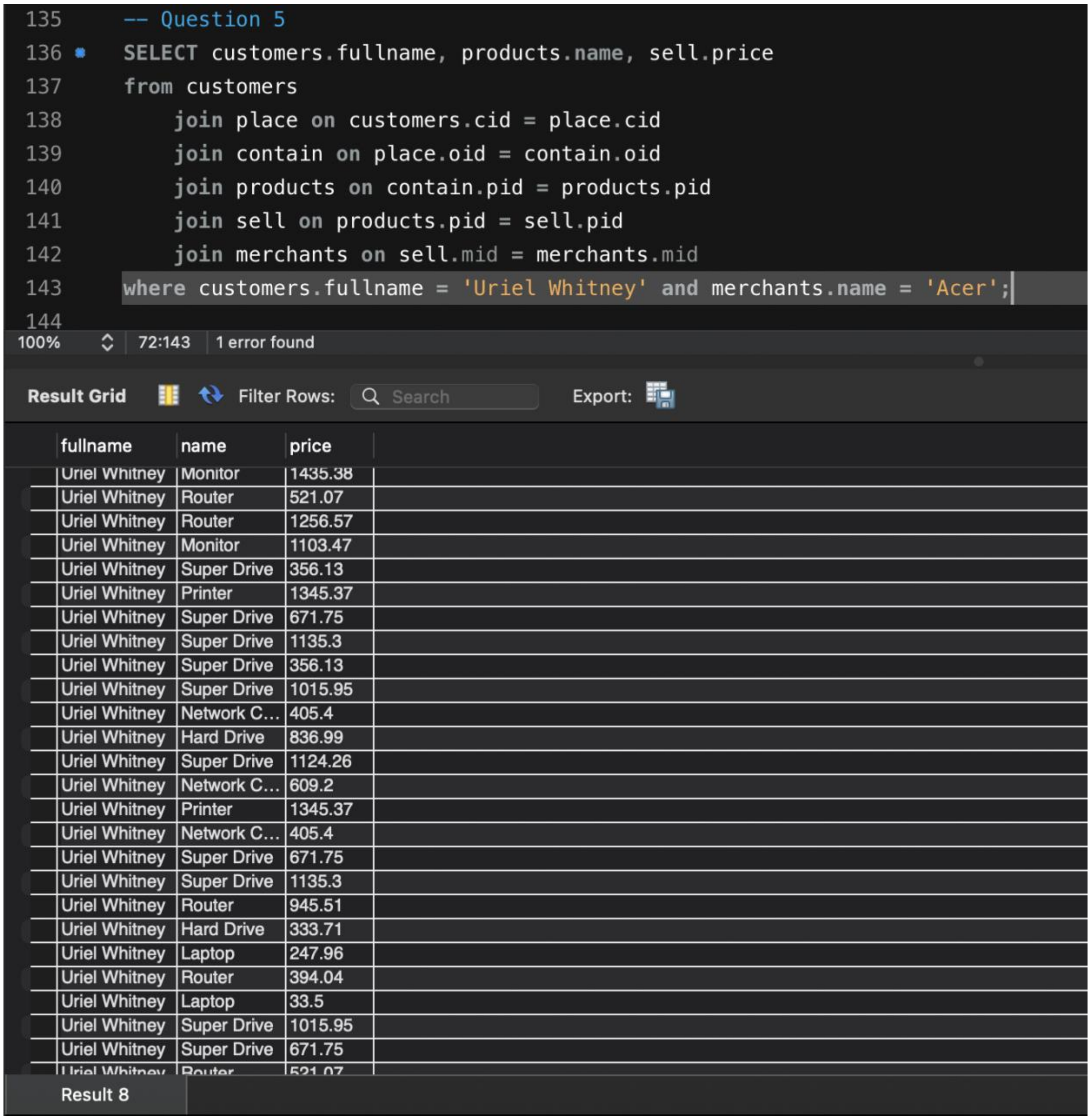
```
-- Question 4
START TRANSACTION; -- allows for resetting the prices back to full price incase query was wrong
    UPDATE sell
        join merchants on merchants.mid = sell.mid
        join products on products.pid = sell.pid
    SET sell.price = (sell.price * .8)
    WHERE merchants.name = 'HP' AND products.category = 'Networking';

    SELECT sell.mid, sell.pid, merchants.name, products.category, sell.price
        from sell
        join merchants on sell.mid = merchants.mid
        join products on products.pid = sell.pid
    where merchants.name = 'HP' and products.category = 'Networking';
ROLLBACK; -- incase update goes wrong

100%    67:132    1 error found
```

mid	pid	name	category	price
3	8	HP	Networking	827.5680000000001
3	10	HP	Networking	923.7440000000001
3	12	HP	Networking	276.008
3	13	HP	Networking	209.76
3	16	HP	Networking	1008.3600000000001
3	18	HP	Networking	164.448
3	19	HP	Networking	1179.896
3	20	HP	Networking	441.616
3	23	HP	Networking	80.76
3	28	HP	Networking	943.2080000000001

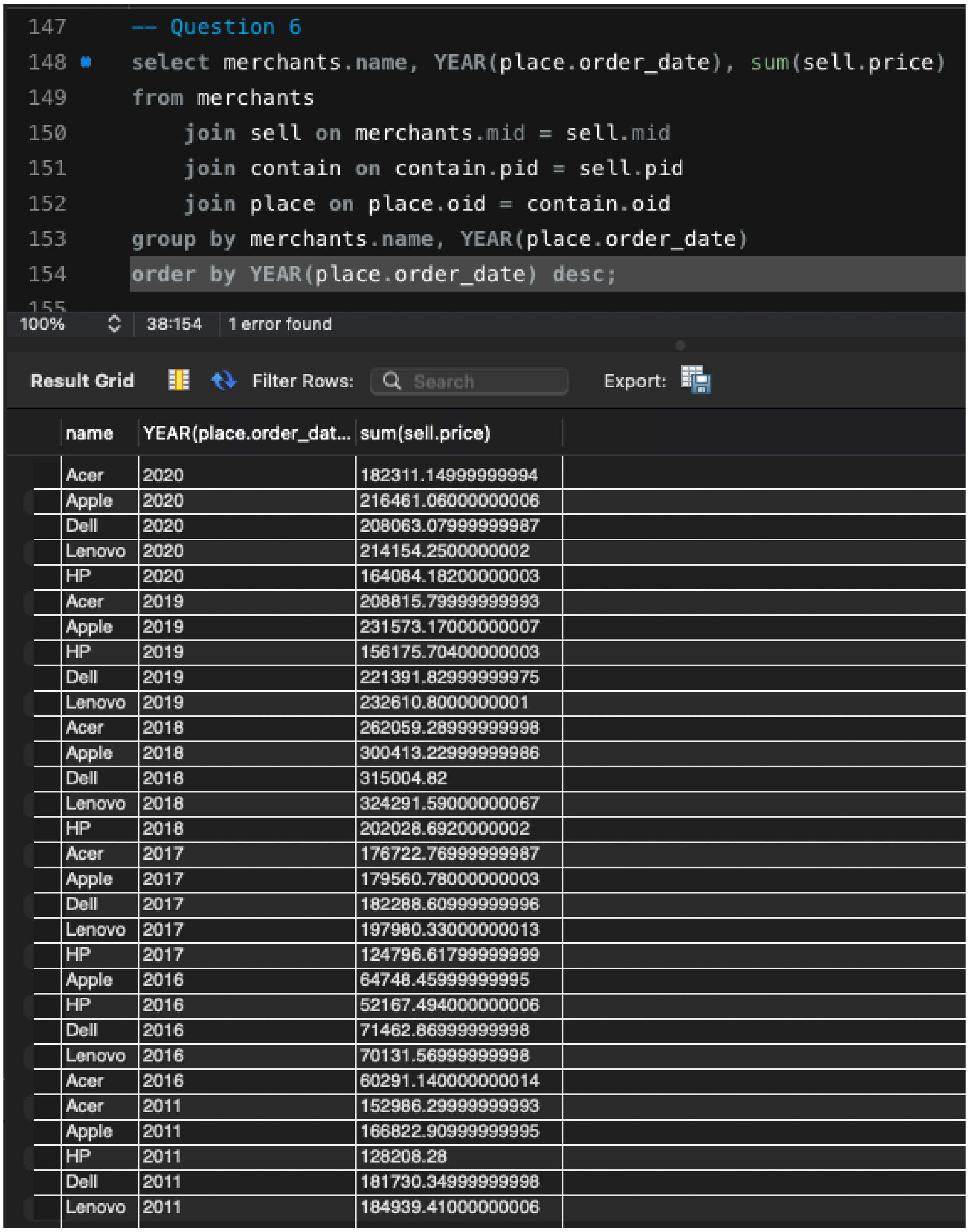
The fourth query aims to provide a 20% discount for every HP networking product. To achieve this, an update query is executed where the price in the sell table is multiplied by .8 to set the new price to be 80% of the original price. In order to ensure only the HP networking products are affected by this, sell is joined with merchants and products using an inner join on the id attributes. This allows the where statement to access the merchant name attribute and product category to make sure they are both equal to HP and networking respectively. To show these changes in the output table, the same joins and where statement are executed again in a select statement where the merchant id, product id, merchant name, product category, and new product price are selected to be columns in the output table. Finally the start transaction and rollback keywords surround the statements. The start transaction keyword makes it so that the original table data is saved somewhere incase an update goes wrong. If the table data needs to be reset to the original data, the rollback statement can be executed.



```sql
-- Question 5
SELECT customers.fullname, products.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';
```

fullname	name	price
Uriel Whitney	Monitor	1435.38
Uriel Whitney	Router	521.07
Uriel Whitney	Router	1256.57
Uriel Whitney	Monitor	1103.47
Uriel Whitney	Super Drive	356.13
Uriel Whitney	Printer	1345.37
Uriel Whitney	Super Drive	671.75
Uriel Whitney	Super Drive	1135.3
Uriel Whitney	Super Drive	356.13
Uriel Whitney	Super Drive	1015.95
Uriel Whitney	Network C...	405.4
Uriel Whitney	Hard Drive	836.99
Uriel Whitney	Super Drive	1124.26
Uriel Whitney	Network C...	609.2
Uriel Whitney	Printer	1345.37
Uriel Whitney	Network C...	405.4
Uriel Whitney	Super Drive	671.75
Uriel Whitney	Super Drive	1135.3
Uriel Whitney	Router	945.51
Uriel Whitney	Hard Drive	333.71
Uriel Whitney	Laptop	247.96
Uriel Whitney	Router	394.04
Uriel Whitney	Laptop	33.5
Uriel Whitney	Super Drive	1015.95
Uriel Whitney	Super Drive	671.75
Uriel Whitney	Router	521.07

The fifth query aims to list every retrieve the orders Uriel Whitney placed from Acer. To achieve this, customers, place, contain, products, sell, and merchants are joined using an inner join on the id attributes. Using this new combined table, the customers full name, product name and product price is selected to be in the output table. To narrow down the data in the output table, the where statement selects on the rows that have the customer full name listed as Uriel Whitney and the merchant name listed as Acer.



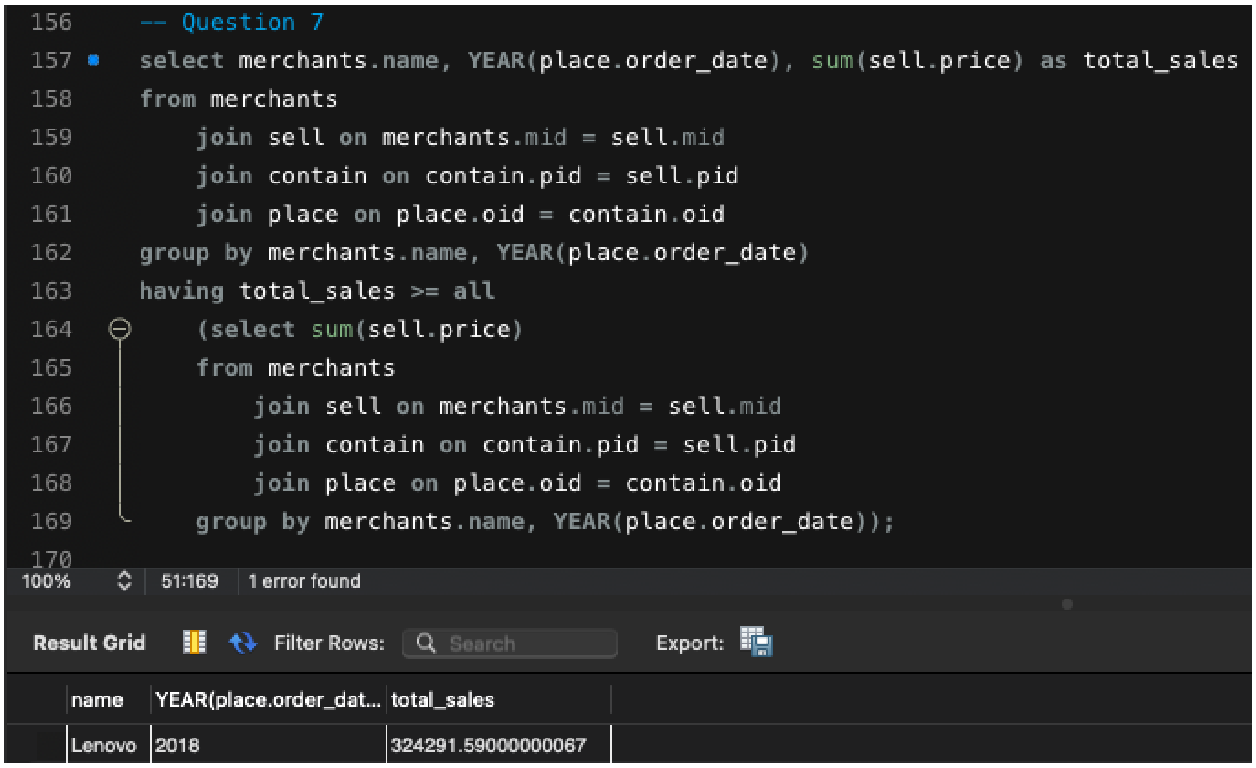
```sql
-- Question 6
select merchants.name, YEAR(place.order_date), sum(sell.price)
from merchants
    join sell on merchants.mid = sell.mid
    join contain on contain.pid = sell.pid
    join place on place.oid = contain.oid
group by merchants.name, YEAR(place.order_date)
order by YEAR(place.order_date) desc;
```

100% 38:154 1 error found

Result Grid Filter Rows: Search Export:

name	YEAR(place.order_dat...	sum(sell.price)
Acer	2020	182311.14999999994
Apple	2020	216461.06000000006
Dell	2020	208063.07999999987
Lenovo	2020	214154.2500000002
HP	2020	164084.18200000003
Acer	2019	208815.79999999993
Apple	2019	231573.17000000007
HP	2019	156175.70400000003
Dell	2019	221391.82999999975
Lenovo	2019	232610.8000000001
Acer	2018	262059.28999999998
Apple	2018	300413.22999999986
Dell	2018	315004.82
Lenovo	2018	324291.59000000067
HP	2018	202028.6920000002
Acer	2017	176722.76999999987
Apple	2017	179560.78000000003
Dell	2017	182288.60999999996
Lenovo	2017	197980.33000000013
HP	2017	124796.61799999999
Apple	2016	64748.45999999995
HP	2016	52167.494000000006
Dell	2016	71462.86999999998
Lenovo	2016	70131.56999999998
Acer	2016	60291.140000000014
Acer	2011	152986.29999999993
Apple	2011	166822.90999999995
HP	2011	128208.28
Dell	2011	181730.34999999998
Lenovo	2011	184939.41000000006

The sixth query aims to list the total annual sales for each company. This is achieved by selecting the merchant name, year from the order date attribute from the order table, and summing the price of the products sold. In order to be able to get all this information, an inner join using the id attribute is done on the merchants, sell, contain, and place tables. To ensure the product prices are added correctly according to each company and year, a group by statement specifies that the sum should be grouped according to merchant name and the year in the order date attribute. Finally, to make the output table easily readable, the output table is ordered by the year in descending order to show the most recent results at the top.

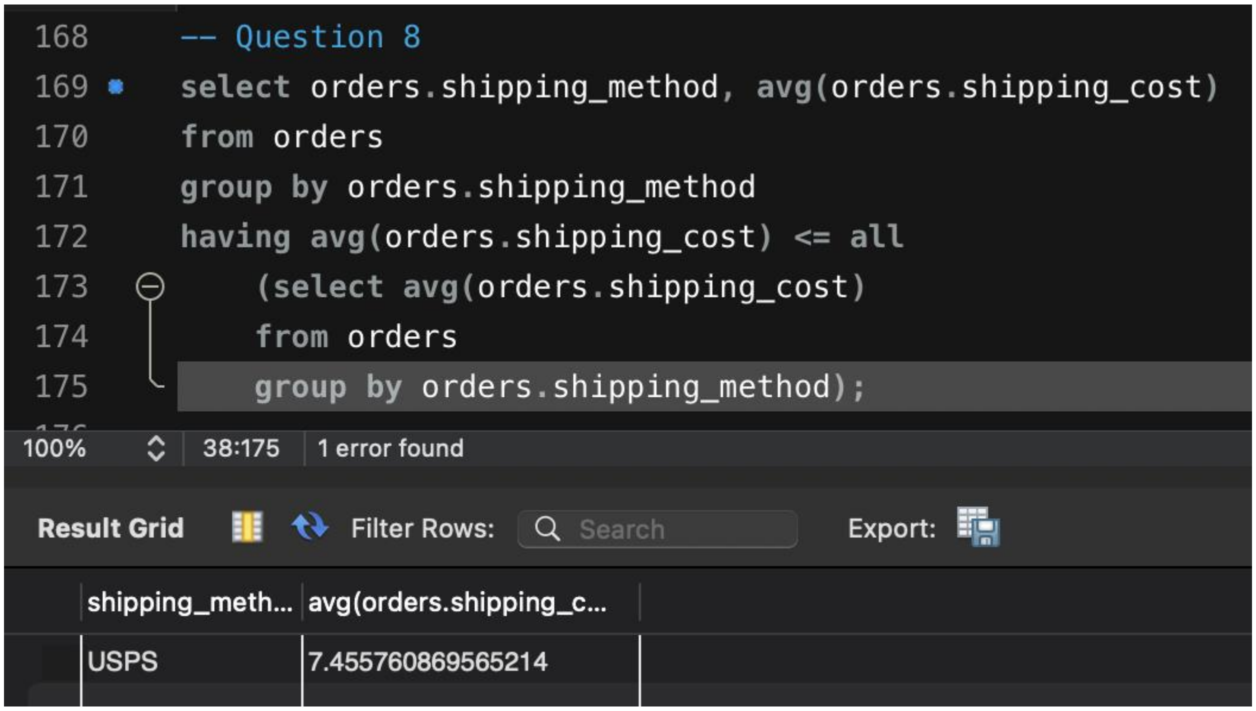


```
-- Question 7
select merchants.name, YEAR(place.order_date), sum(sell.price) as total_sales
from merchants
    join sell on merchants.mid = sell.mid
    join contain on contain.pid = sell.pid
    join place on place.oid = contain.oid
group by merchants.name, YEAR(place.order_date)
having total_sales >= all
    (select sum(sell.price)
    from merchants
        join sell on merchants.mid = sell.mid
        join contain on contain.pid = sell.pid
        join place on place.oid = contain.oid
    group by merchants.name, YEAR(place.order_date));

100%    51:169    1 error found
```

name	YEAR(place.order_dat...	total_sales
Lenovo	2018	324291.59000000067

The seventh query aims to determine which company had the highest annual revenue and in what year that occurred. The same query besides the order by statement is used to begin this query. To distinguish this query from the sixth one a having statement is included. The having statement selects the result from the sixth query that has the highest total sales through a subquery. The subquery determines the sum of the prices of the products sold for each company in each year. All these results are then compared through the having statement and only the results that are greater than or equal to all the other entries are added to the output table. This is similar to using max() but allows for multiple results to be returned if two or more results held the same maximum value.

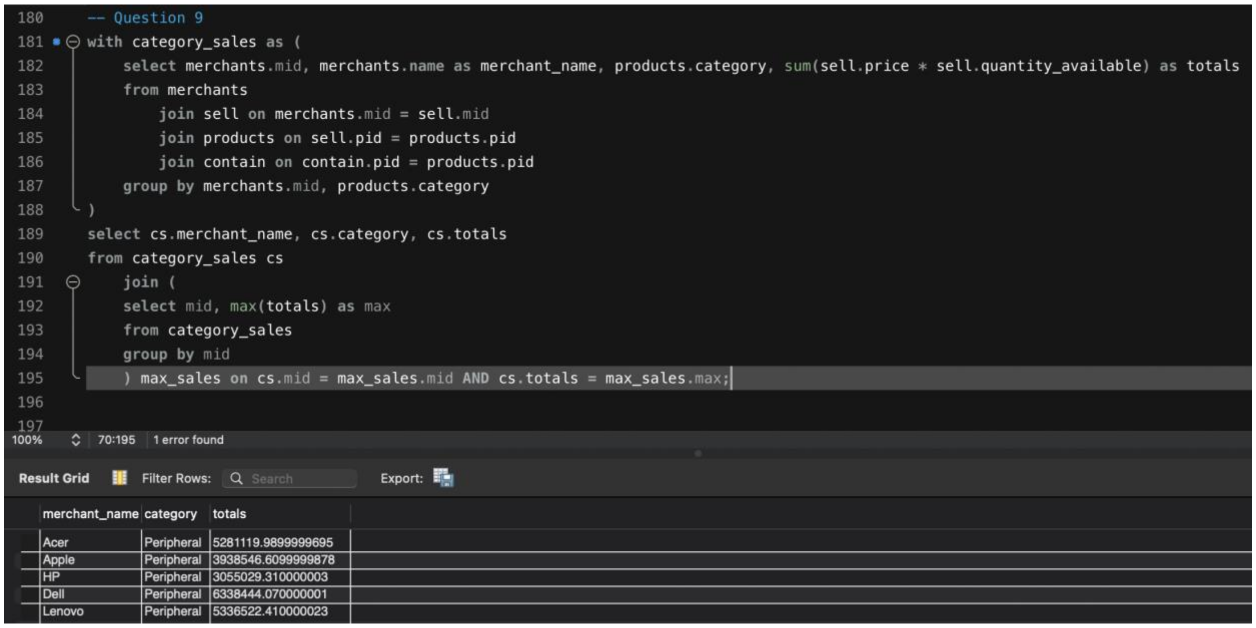


```
-- Question 8
select orders.shipping_method, avg(orders.shipping_cost)
from orders
group by orders.shipping_method
having avg(orders.shipping_cost) <= all
    (select avg(orders.shipping_cost)
    from orders
    group by orders.shipping_method);
```

```
100%   38:175   1 error found
```

shipping_meth...	avg(orders.shipping_c...
USPS	7.455760869565214

The eighth query aims to determine the average cheapest shipping method. This is done by selecting the shipping method and performing the average of the shipping cost for that shipping method. The average is grouped by shipping method to ensure that the average is computed for each shipping method and not the average of all shipping methods combined. Since the shipping method and cost is only available through the orders table and the rows in the orders table only exist if an order is placed, no joins need to be performed to make sure only shipping methods that have actually been used are included in the calculation. To output only the cheapest method, a having statement is executed where the average shipping cost grouped by shipping method is again computed in a subquery. Then the results from that are compared to one another and the having statement makes it so that only the minimum value is outputted to the output table. If multiple equal minimum values exist, all of those will be provided in the output table.

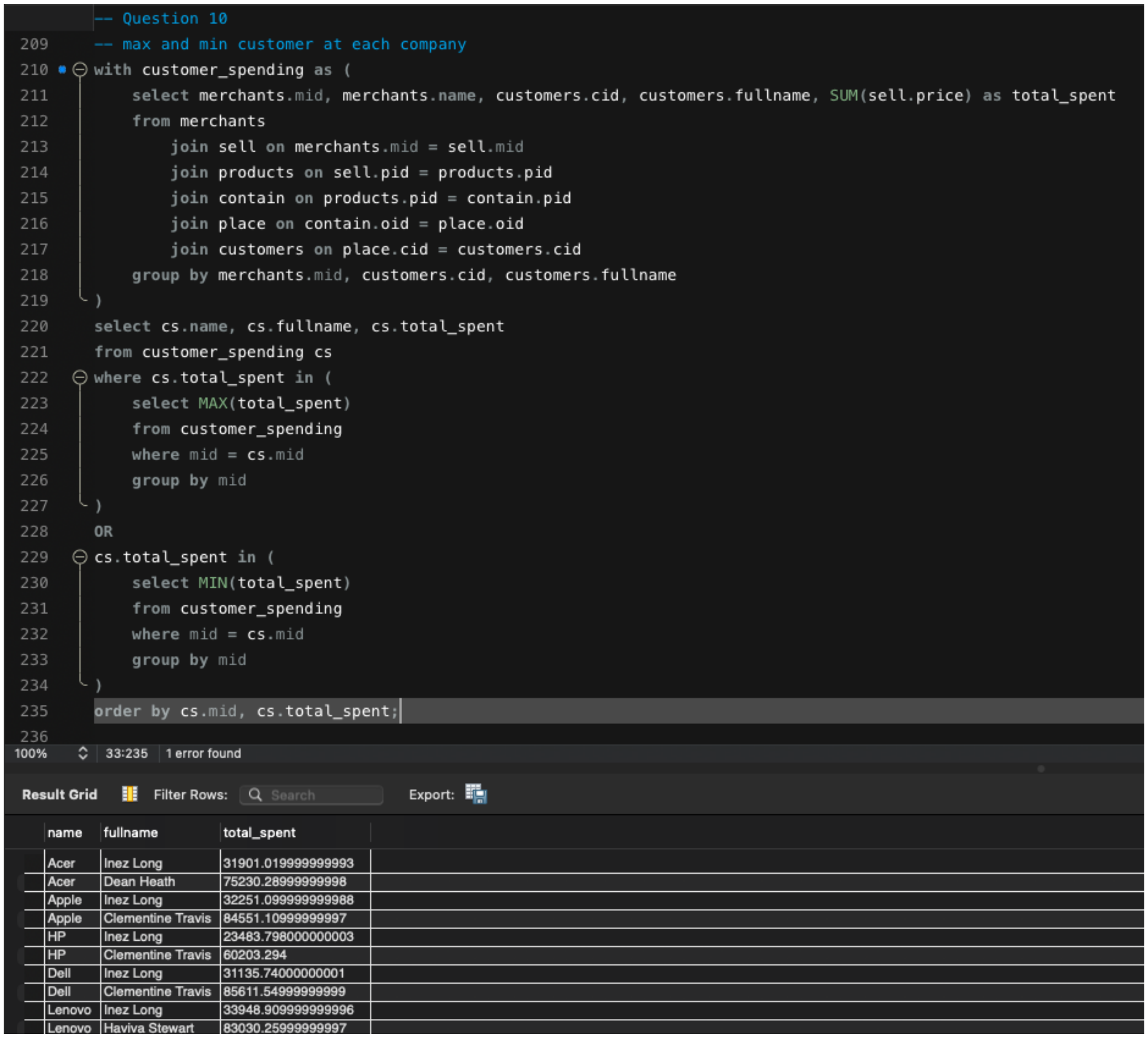


```sql
-- Question 9
with category_sales as (
    select merchants.mid, merchants.name as merchant_name, products.category, sum(sell.price * sell.quantity_available) as totals
    from merchants
        join sell on merchants.mid = sell.mid
        join products on sell.pid = products.pid
        join contain on contain.pid = products.pid
    group by merchants.mid, products.category
)
select cs.merchant_name, cs.category, cs.totals
from category_sales cs
    join (
    select mid, max(totals) as max
    from category_sales
    group by mid
    ) max_sales on cs.mid = max_sales.mid AND cs.totals = max_sales.max;
```

```
100%    70:195    1 error found
```

merchant_name	category	totals
Acer	Peripheral	5281119.9899999695
Apple	Peripheral	3938546.6099999878
HP	Peripheral	3055029.310000003
Dell	Peripheral	6338444.070000001
Lenovo	Peripheral	5336522.410000023

The ninth query aims to determine what the best sold category for each company is based on the amount of money that category has produced for the company in sales. To accomplish this, first a CTE table is created as category sales. This table selects the merchant id and name, product category, and the sum of the price of products sold. The merchant name receives an alias of merchant_name and the sum receives an alias of totals. This select statement is possible through the inner join between merchants, sell, products, and contain on the id attributes. The sum is calculated for each category for each merchant through the group by statement, and the results of this output table are saved as the category sales table to be used in the main query. The main query selects the merchant name, category, and totals from the columns of the CTE table. It is then joined with a table formed through a subquery. The subquery selects the merchant id and the maximum of the totals for each merchant. The tables from the CTE and subquery are then joined using an inner join where the merchant ids are the same and the sum from the CTE table and the max total returned from the subquery are both the same. This ensures that each company is listed and only their highest selling category is included in the output table.



```sql
-- Question 10
-- max and min customer at each company
with customer_spending as (
    select merchants.mid, merchants.name, customers.cid, customers.fullname, SUM(sell.price) as 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 place on contain.oid = place.oid
        join customers on place.cid = customers.cid
    group by merchants.mid, customers.cid, customers.fullname
)
select cs.name, cs.fullname, cs.total_spent
from customer_spending cs
where cs.total_spent in (
    select MAX(total_spent)
    from customer_spending
    where mid = cs.mid
    group by mid
)
OR
cs.total_spent in (
    select MIN(total_spent)
    from customer_spending
    where mid = cs.mid
    group by mid
)
order by cs.mid, cs.total_spent;
```

```
100%    33:235    1 error found
```

name	fullname	total_spent
Acer	Inez Long	31901.019999999993
Acer	Dean Heath	75230.28999999998
Apple	Inez Long	32251.099999999988
Apple	Clementine Travis	84551.10999999997
HP	Inez Long	23483.798000000003
HP	Clementine Travis	60203.294
Dell	Inez Long	31135.74000000001
Dell	Clementine Travis	85611.54999999999
Lenovo	Inez Long	33948.909999999996
Lenovo	Haviva Stewart	83030.25999999997

The tenth query aims to find out what customers have spent the most and least amounts for each company. The query begins with creating a CTE table named customer spending. This table selects the merchant id and name, customer id and name, and finds the sum of the prices of products sold under the alias total_spent. These selects are possible through inner joins on the id attribute between the merchants, sell, products, contain, place, and customers tables. The sum is computed individually for each company and customer through the group by statement which specifies that the sums should be grouped by company first and then by customer. After this table is created with the attributes in the select statement as the columns, the main query begins by selecting the merchant name, customer name, and total spent columns from the CTE table. The rows returned from the select statement are then narrowed down to include only the maximum and minimum spenders through the where statement in the main query. The max is calculated first through a subquery in the where statement. This subquery selects the max entry in total spent column from the CTE table where the merchant id matches in the subquery and the CTE table. This is done for each company through the group by statement that specifies it should be grouped together by merchant id. The same process in then followed in a second subquery in the where statement for the minimum value. Both of these values are returned in the output table for each company through the or keyword in the where statement between the subqueries. Finally, the output table is ordered by the minimum then maximum value for each company, making sure that each company's results are outputted in subsequent rows together.