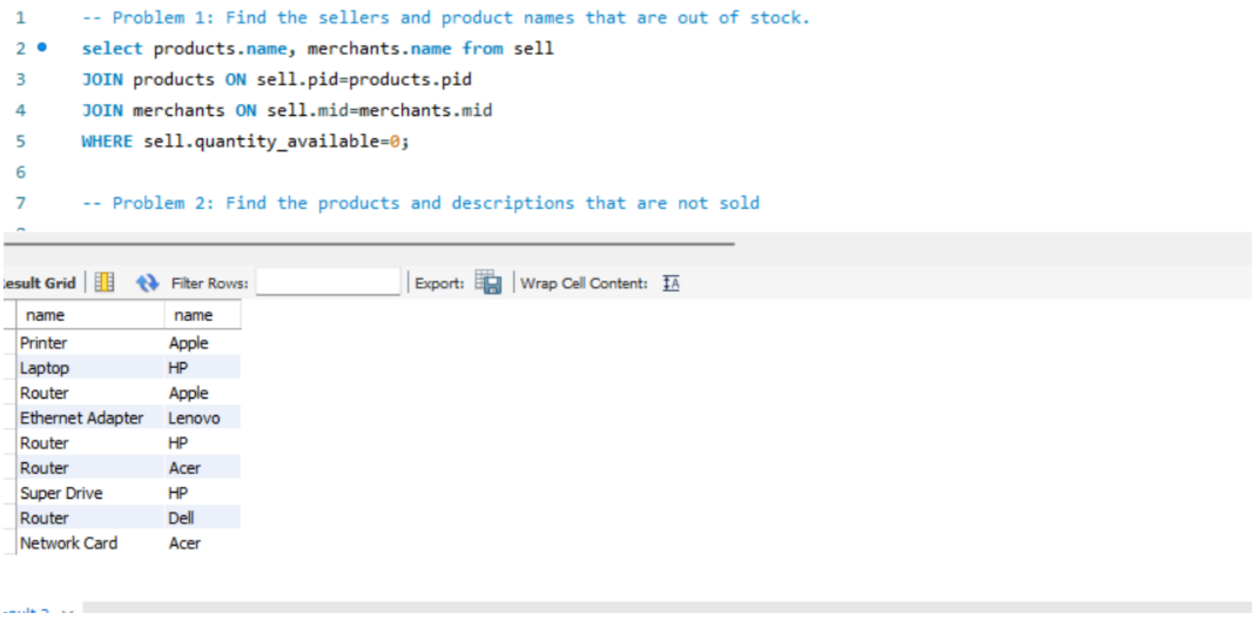
Assumptions for code: every oid in contain represents the order of one item. shipping cost only affects the consumer, and is not factored into company based queries.

Problem 1: Find the sellers and product names that are out of stock.
Query:
select products.name, merchants.name from sell
JOIN products ON sell.pid=products.pid
JOIN merchants ON sell.mid=merchants.mid
WHERE sell.quantity_available=0;

Explanation:
Selects the products name and merchants name, natural joins based on pid and mid, and lastly provides a condition that the quantity available property is zero.

Screenshot:



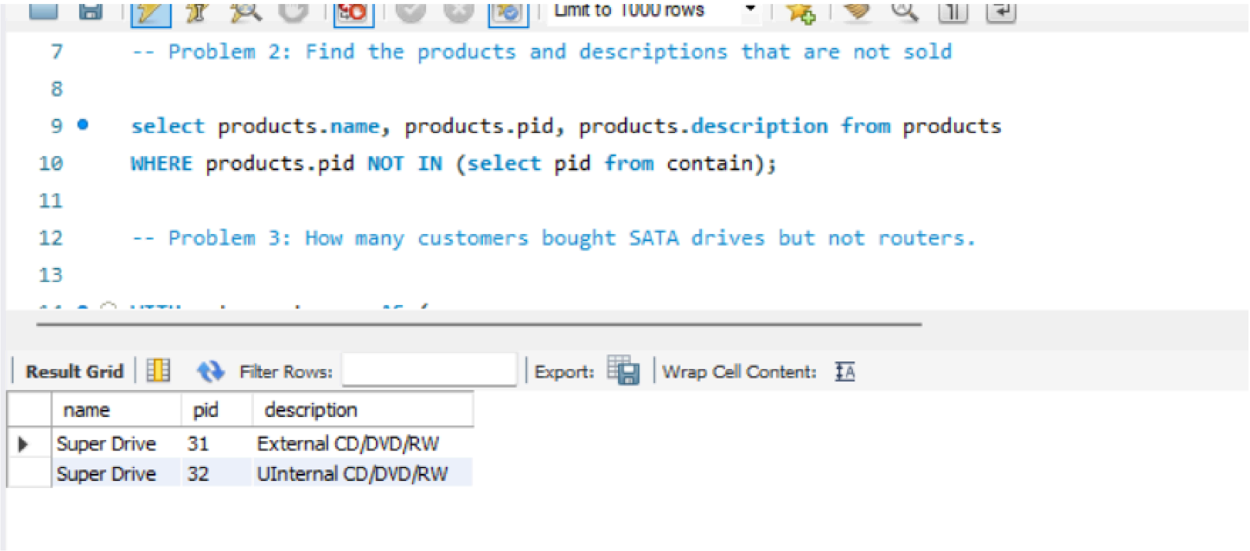
```
-- Problem 1: Find the sellers and product names that are out of stock.
select products.name, merchants.name from sell
JOIN products ON sell.pid=products.pid
JOIN merchants ON sell.mid=merchants.mid
WHERE sell.quantity_available=0;

-- Problem 2: Find the products and descriptions that are not sold
```

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

Problem 2 :Find the products and descriptions that are not sold
Query:
select products.name, products.pid, products.description from products
where products.pid NOT IN (select pid from contain);

Explanation:
Selects name, pid, and description from the products. Condition is where the product id is not in the subquery select where pid is in contain.

Screenshot:



```
-- Problem 2: Find the products and descriptions that are not sold

select products.name, products.pid, products.description from products
WHERE products.pid NOT IN (select pid from contain);

-- Problem 3: How many customers bought SATA drives but not routers.
```

name	pid	description
Super Drive	31	External CD/DVD/RW
Super Drive	32	UInternal CD/DVD/RW

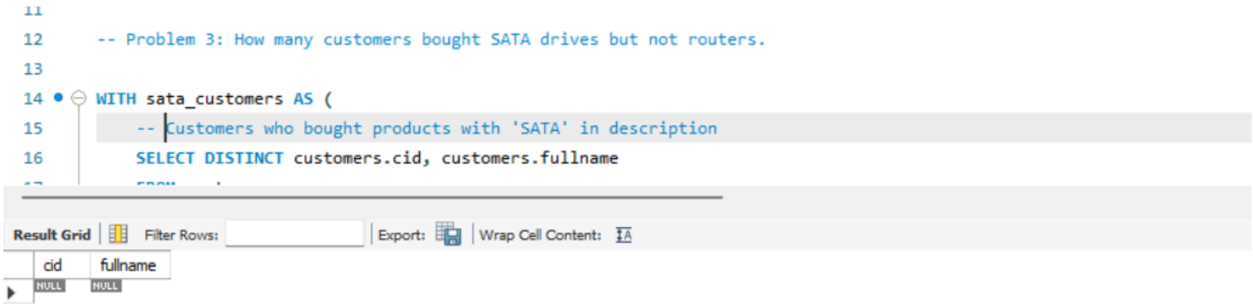
Problem 3: How many customers bought SATA drives but not routers.

Query:

```
WITH sata_customers AS (
        SELECT DISTINCT customers.cid, customers.fullname
    FROM customers
    JOIN place ON customers.cid = place.cid
    JOIN contain ON place.oid = contain.oid
    JOIN products ON contain.pid = products.pid
    WHERE products.description LIKE '%SATA%'
),
router_customers AS (
        SELECT DISTINCT customers.cid
    FROM customers
    JOIN place ON customers.cid = place.cid
    JOIN contain ON place.oid = contain.oid
    JOIN products ON contain.pid = products.pid
    WHERE products.name = 'Router'
)

SELECT DISTINCT sata_customers.cid, sata_customers.fullname
FROM sata_customers
LEFT JOIN router_customers ON sata_customers.cid = router_customers.cid
WHERE router_customers.cid IS NULL
UNION ALL
SELECT NULL, NULL
WHERE NOT EXISTS (
    SELECT 1
    FROM sata_customers
    LEFT JOIN router_customers ON sata_customers.cid = router_customers.cid
    WHERE router_customers.cid IS NULL);
```

Explanation:
The query first creates a new table via the With statement called sata_customers. This table displays which customers bought SATA description products. Distinct keywords are used so that a single customer will not appear multiple times. Joins on cid, oid, pid. The where statement looks for any description with the word 'SATA'. A second table is also created, router_customers, which displays the information of which customers bought products of the category 'router'. Joins on pid, cid, and oid. The where statement looks for products that were purchased using in the 'router' category. The second to last select statement selects from the sata customers their id and full name, with left join on the router table's and sata table's cid attribute and so that all router values will be null if the customer did not buy a router. The where statement makes it so that only values that are null in the router purchase table will be displayed. If none exist, the last query will display a row of null values.

Screenshot:



```
-- Problem 3: How many customers bought SATA drives but not routers.

WITH sata_customers AS (
    -- Customers who bought products with 'SATA' in description
    SELECT DISTINCT customers.cid, customers.fullname
```

cid	fullname
NULL	NULL

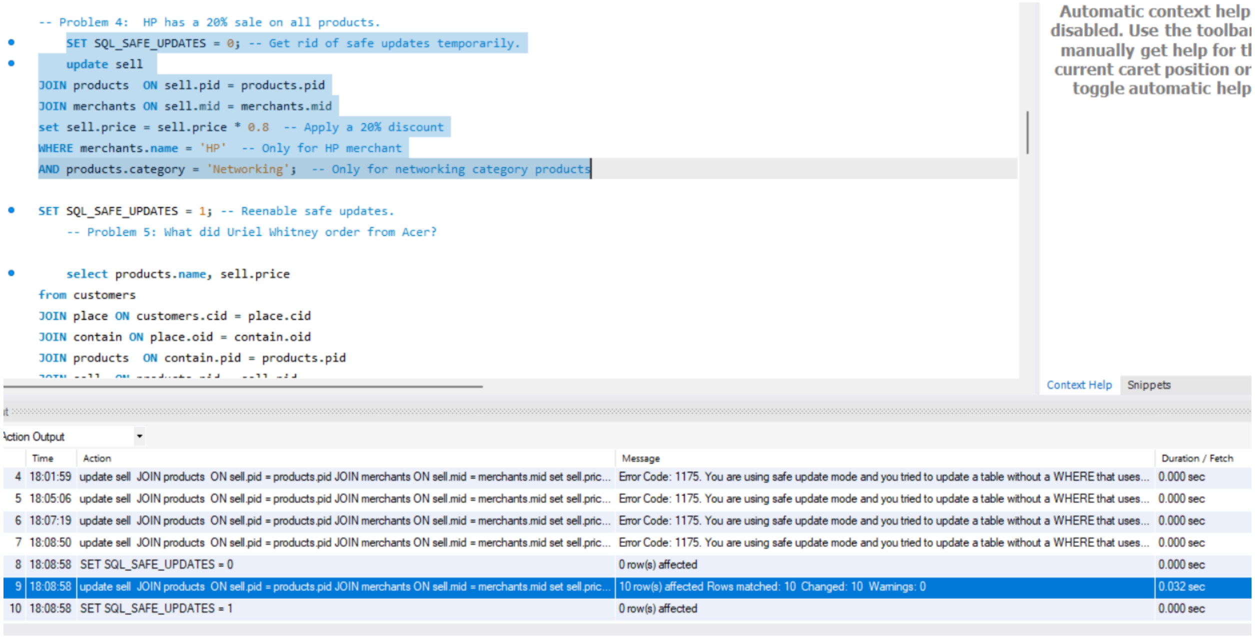
Problem 4: HP is having a 20% sale on networking products

Query:

```
update sell
JOIN products  ON sell.pid = products.pid
JOIN merchants ON sell.mid = merchants.mid
set sell.price = sell.price * 0.8  -- Apply a 20% discount
WHERE merchants.name = 'HP'  -- Only for HP merchant
AND products.category = 'Networking';  -- Only for networking category products
```

Explanation:
Query first updates the sell table, joins sell, merchants, and products on pid and mid to reference name and category. Sets price to 80% of what it was before to represent a 20% discount, specifically in cases where merchant name is HP and the category is Networking.

Screenshot:



```
-- Problem 4:  HP has a 20% sale on all products.
    SET SQL_SAFE_UPDATES = 0; -- Get rid of safe updates temporarily.
    update sell
JOIN products  ON sell.pid = products.pid
JOIN merchants ON sell.mid = merchants.mid
set sell.price = sell.price * 0.8  -- Apply a 20% discount
WHERE merchants.name = 'HP'  -- Only for HP merchant
AND products.category = 'Networking';  -- Only for networking category products

SET SQL_SAFE_UPDATES = 1; -- Reenable safe updates.
    -- Problem 5: What did Uriel Whitney order from Acer?

    select 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
```

Automatic context help
disabled. Use the toolbai
manually get help for tl
current caret position or
toggle automatic help

	Time	Action	Message	Duration / Fetch
4	18:01:59	update sell JOIN products ON sell.pid = products.pid JOIN merchants ON sell.mid = merchants.mid set sell.pric...	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses...	0.000 sec
5	18:05:06	update sell JOIN products ON sell.pid = products.pid JOIN merchants ON sell.mid = merchants.mid set sell.pric...	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses...	0.000 sec
6	18:07:19	update sell JOIN products ON sell.pid = products.pid JOIN merchants ON sell.mid = merchants.mid set sell.pric...	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses...	0.000 sec
7	18:08:50	update sell JOIN products ON sell.pid = products.pid JOIN merchants ON sell.mid = merchants.mid set sell.pric...	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses...	0.000 sec
8	18:08:58	SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.000 sec
9	18:08:58	update sell JOIN products ON sell.pid = products.pid JOIN merchants ON sell.mid = merchants.mid set sell.pric...	10 row(s) affected Rows matched: 10 Changed: 10 Warnings: 0	0.032 sec
10	18:08:58	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.000 sec

Problem 5 :What did Uriel Whitney order from Acer?

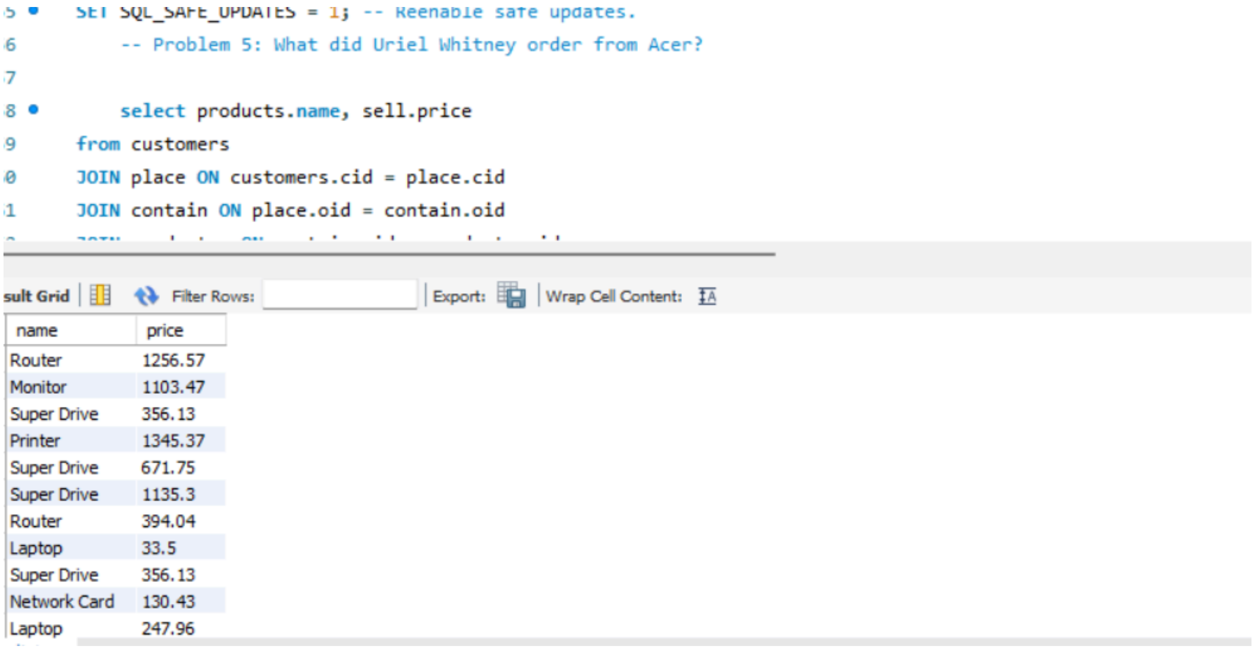
Query:

```
 SELECT 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';
```

Explanation:

Selects names and prices from products and sell. Natural joins on cid, oid, pid, and mid. Where statement checks for Uriel Whitney’s full name and checks for the merchants name Acer.

Screenshot:



```
SET SQL_SAFE_UPDATES = 1; -- Reenable safe updates.
    -- Problem 5: What did Uriel Whitney order from Acer?

    select products.name, sell.price
from customers
JOIN place ON customers.cid = place.cid
JOIN contain ON place.oid = contain.oid
```

name	price
Router	1256.57
Monitor	1103.47
Super Drive	356.13
Printer	1345.37
Super Drive	671.75
Super Drive	1135.3
Router	394.04
Laptop	33.5
Super Drive	356.13
Network Card	130.43
Laptop	247.96

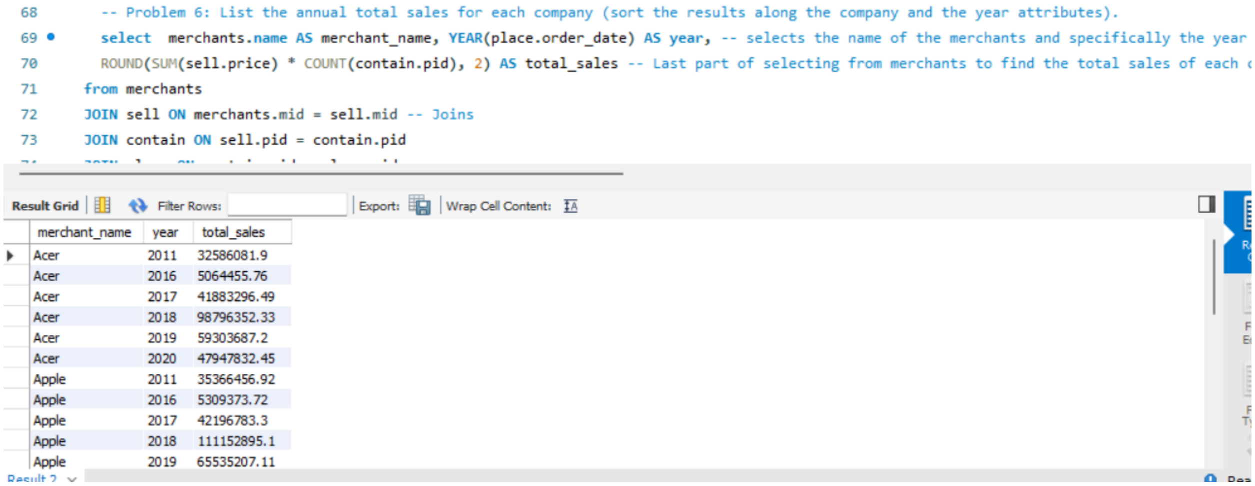
Problem 6:List the annual total sales for each company

Query:

```
select  merchants.name AS merchant_name, YEAR(place.order_date) AS year,
ROUND(SUM(sell.price) * COUNT(contain.pid), 2) AS total_sales
from merchants
JOIN sell ON merchants.mid = sell.mid -- Joins
JOIN contain ON sell.pid = contain.pid
JOIN place ON contain.oid = place.oid
JOIN orders ON place.oid = orders.oid
GROUP BY merchants.name, year
ORDER BY merchants.name ASC, year ASC;
```

Explanation: The query first selects the merchant name, uses the year function to extract the date from the order date, and then finds the price of the product times the number of times its pid appears in contain, representing it being ordered. The values are then all summed together and rounded to 2 decimal places, for readability. Natural joins are made on mid, pid, and oid. Results are grouped by company name and year, and are ordered in ascending order for the company, then by year, so that the values are all organized first based on name of the company so they’re all in a row, and then by year so that it is a nice and neat timeline.

Screenshot:



```
    -- Problem 6: List the annual total sales for each company (sort the results along the company and the year attributes).
    select  merchants.name AS merchant_name, YEAR(place.order_date) AS year, -- selects the name of the merchants and specifically the year
    ROUND(SUM(sell.price) * COUNT(contain.pid), 2) AS total_sales -- Last part of selecting from merchants to find the total sales of each c
  from merchants
  JOIN sell ON merchants.mid = sell.mid -- Joins
  JOIN contain ON sell.pid = contain.pid
```

merchant_name	year	total_sales
Acer	2011	32586081.9
Acer	2016	5064455.76
Acer	2017	41883296.49
Acer	2018	98796352.33
Acer	2019	59303687.2
Acer	2020	47947832.45
Apple	2011	35366456.92
Apple	2016	5309373.72
Apple	2017	42196783.3
Apple	2018	111152895.1
Apple	2019	65535207.11

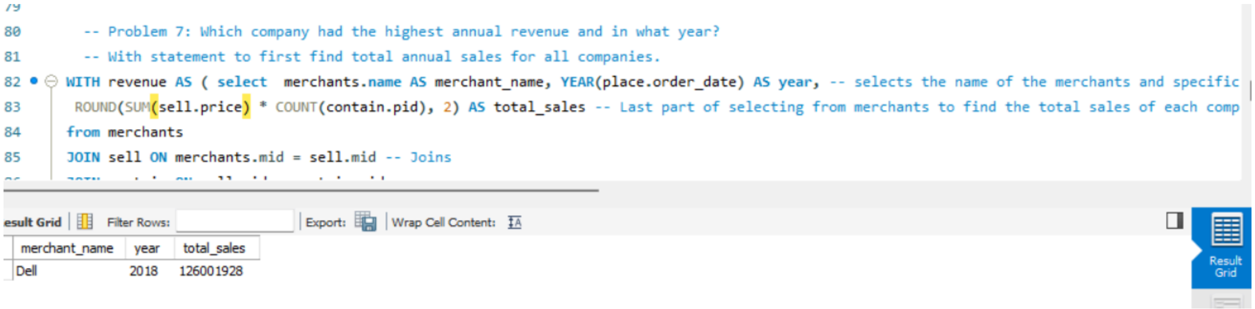
Problem 7:Which company had the highest annual revenue and in what year?

Query:

WITH revenue AS (select merchants.name AS merchant_name, YEAR(place.order_date) AS year, ROUND(SUM(sell.price) * COUNT(contain.pid), 2) 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
JOIN orders ON place.oid = orders.oid
GROUP BY merchants.name, year)
select merchant_name, year, total_sales
from revenue
WHERE total_sales = (select MAX(total_sales) from revenue);

Explanation: First the with statement is used to create a temporary table query. The query first selects the merchant name, uses the year function to extract the date from the order date, and then finds the price of the product times the number of times its pid appears in contain, representing it being ordered. The values are then all summed together and rounded to 2 decimal places, for readability. Natural joins are made on mid, pid, and oid. Results are grouped by company name and year. Then a second main query is used to select merchant_name, year, and total_sales from revenue, in the condition that total_sales is equal to the maximum value of total_sales from the revenue table.

Screenshot:



```
  -- Problem 7: Which company had the highest annual revenue and in what year?
  -- With statement to first find total annual sales for all companies.
WITH revenue AS ( select  merchants.name AS merchant_name, YEAR(place.order_date) AS year, -- selects the name of the merchants and specific
 ROUND(SUM(sell.price) * COUNT(contain.pid), 2) AS total_sales -- Last part of selecting from merchants to find the total sales of each comp
from merchants
JOIN sell ON merchants.mid = sell.mid -- Joins
```

merchant_name	year	total_sales
Dell	2018	126001928

Problem 8:On average, what was the cheapest shipping method used ever?

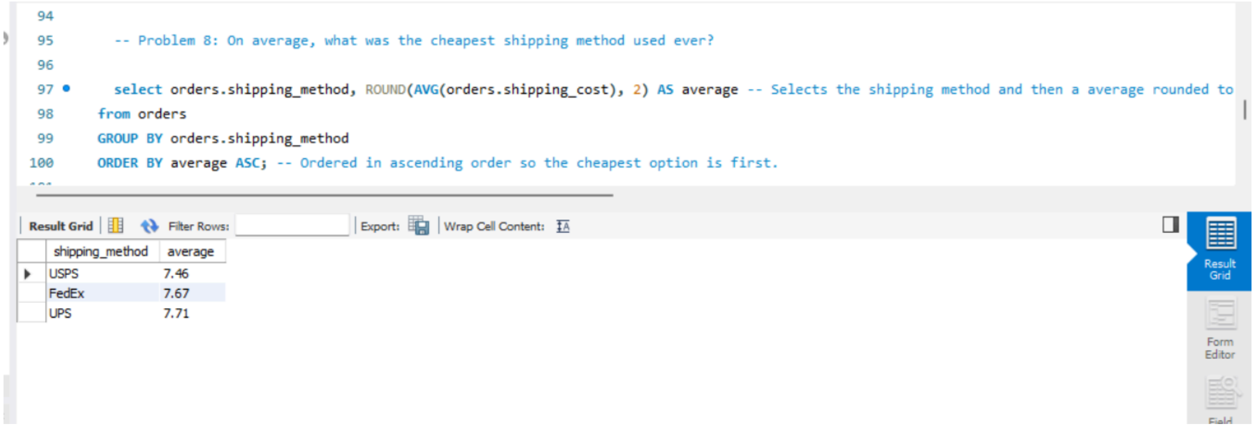
Query:

SELECT orders.shipping_method, AVG(orders.shipping_cost, 2) AS average_cost
FROM orders
GROUP BY orders.shipping_method
ORDER BY average_cost ASC;

Explanation:

Selects from orders on shipping method, finds the average shipping cost of the shipping cost section, limited to 2 decimal places for readability, and is ordered in ascending order and grouped by shipping method.

Screenshot:



```

  -- Problem 8: On average, what was the cheapest shipping method used ever?

  select orders.shipping_method, ROUND(AVG(orders.shipping_cost), 2) AS average -- Selects the shipping method and then a average rounded to
from orders
GROUP BY orders.shipping_method
ORDER BY average ASC; -- Ordered in ascending order so the cheapest option is first.
```

shipping_method	average
USPS	7.46
FedEx	7.67
UPS	7.71

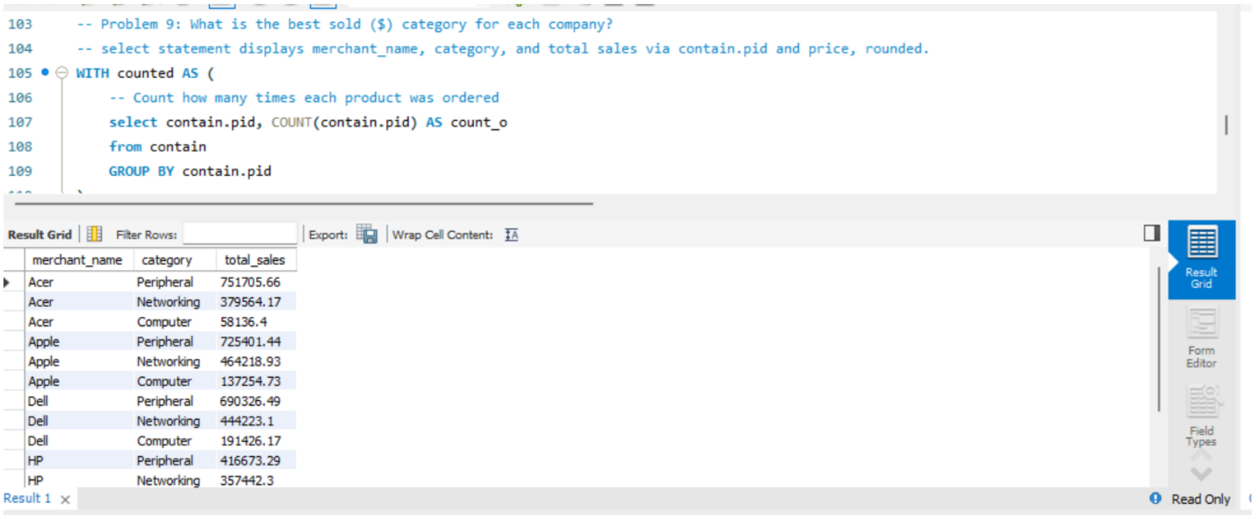
Problem 9:What is the best sold ($) category for each company?

Query:

```
WITH counted AS (
     select contain.pid, COUNT(contain.pid) AS count_o
    from contain
    GROUP BY contain.pid
)
select merchants.name AS merchant_name,
       products.category,
       ROUND(SUM(sell.price * counted.count_o), 2) AS total_sales
from merchants
JOIN sell ON merchants.mid = sell.mid
JOIN products ON sell.pid = products.pid
JOIN counted ON sell.pid = counted.pid
GROUP BY merchants.name, products.category
ORDER BY merchants.name, total_sales DESC;
```

Explanation: With statement finds the counted number of products for each category. The query first selects to display merchant name, product category, and then the calculated sum of the price of a product times the number of occurrences of said product's pid to get the total of sales. Natural Joins on pid, mid, and oid. Items are grouped by merchant name and the category, and then ordered in descending order to display which category is the most profitable for each company in monetary value.

Screenshot:



```
-- Problem 9: What is the best sold ($) category for each company?
-- select statement displays merchant_name, category, and total sales via contain.pid and price, rounded.
WITH counted AS (
    -- Count how many times each product was ordered
    select contain.pid, COUNT(contain.pid) AS count_o
    from contain
    GROUP BY contain.pid
```

merchant_name	category	total_sales
Acer	Peripheral	751705.66
Acer	Networking	379564.17
Acer	Computer	58136.4
Apple	Peripheral	725401.44
Apple	Networking	464218.93
Apple	Computer	137254.73
Dell	Peripheral	690326.49
Dell	Networking	444223.1
Dell	Computer	191426.17
HP	Peripheral	416673.29
HP	Networking	357442.3

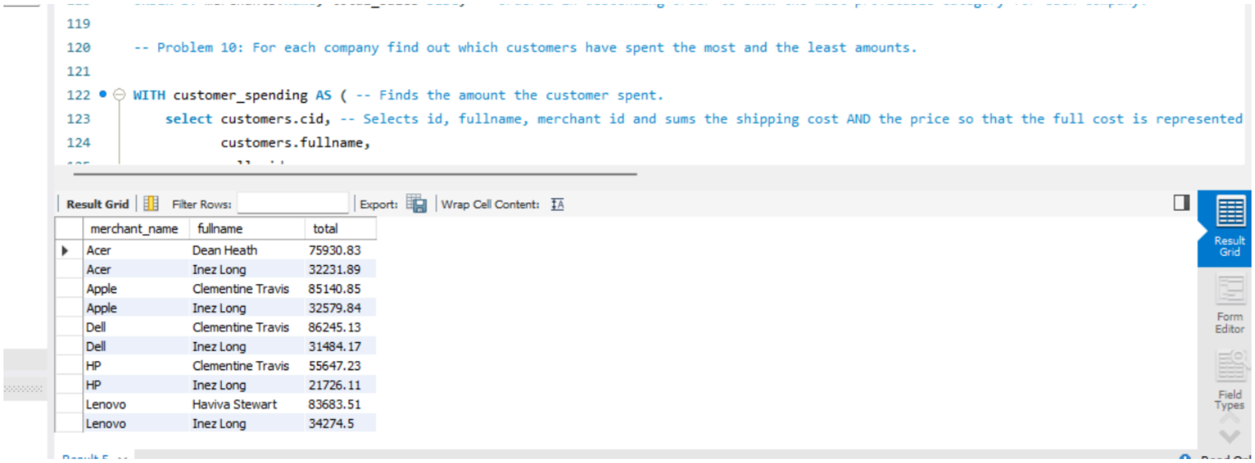
Problem 10: For each company, find out which customers have spent the most and the least amounts.

Query:

```
WITH customer_spending AS (
select customers.cid, customers.fullname, sell.mid, SUM(sell.price + orders.shipping_cost) AS
total
 from customers
    JOIN place ON customers.cid = place.cid
    JOIN contain ON place.oid = contain.oid
    JOIN orders ON place.oid = orders.oid
    JOIN sell ON contain.pid = sell.pid
    GROUP BY customers.cid, customers.fullname, sell.mid),
ranked_customers AS (  select customer_spending.cid, customer_spending.fullname,
customer_spending.mid,
ROUND(customer_spending.total, 2) AS total,

 ROW_NUMBER() OVER (PARTITION BY customer_spending.mid
ORDER BY customer_spending.total DESC) AS rank_highest,

           ROW_NUMBER() OVER (PARTITION BY customer_spending.mid ORDER BY
customer_spending.total ASC) AS rank_lowest
    from customer_spending
)

select merchants.name AS merchant_name,
       ranked_customers.fullname,
       ranked_customers.total
from ranked_customers
JOIN merchants ON ranked_customers.mid = merchants.mid
WHERE ranked_customers.rank_highest = 1 OR ranked_customers.rank_lowest = 1
ORDER BY merchants.name,
         ranked_customers.total DESC;
```

Explanation:
First a With statement is used to create a temporary table of customer spending. A select statement selects customer id, their name, the merchant id, and then finds the sum of all the products they ordered plus the shipping cost of each of those products and then labels it as total. Joins on cid, oid, and pid. Then grouped by cid, fullname, and merchant id. Then a separate table, that ranks the customers. Partitioning and window statements are then used and ordered in descending, and then in ascending order to find the individuals who spent the most and the least. The last select query then selects the merchant’s name, and each of the ranked customer’s full names and their total purchases. Joins then join the tables on mid, and the where statement makes it so that only the highest and lowest are displayed. Then the order by statement shows each of the merchants in order, and then the highest customer first, and then the lowest customer second.

Screenshot:



```sql

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

WITH customer_spending AS ( -- Finds the amount the customer spent.
    select customers.cid, -- Selects id, fullname, merchant id and sums the shipping cost AND the price so that the full cost is represented
           customers.fullname,
```

merchant_name	fullname	total
Acer	Dean Heath	75930.83
Acer	Inez Long	32231.89
Apple	Clementine Travis	85140.85
Apple	Inez Long	32579.84
Dell	Clementine Travis	86245.13
Dell	Inez Long	31484.17
HP	Clementine Travis	55647.23
HP	Inez Long	21726.11
Lenovo	Haviva Stewart	83683.51
Lenovo	Inez Long	34274.5