DB Assignment 3

Giannina Flamiano

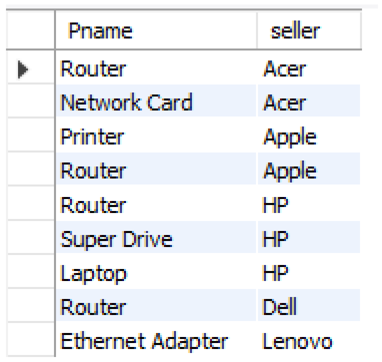
October 11, 2024

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

SQL QUERY

```sql
SELECT p.name AS Pname, m.name AS seller
FROM products p
INNER JOIN sell s ON p.pid = s.pid
INNER JOIN merchants m ON s.mid = m.mid
WHERE s.quantity_available=0;
```

SCREENSHOT



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

EXPLANATION

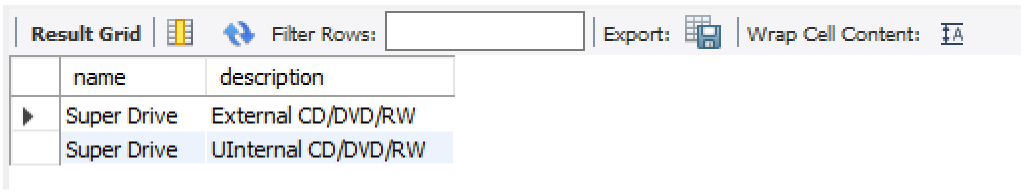
This query looks at products, sell, and merchants tables. It joins the products table to get the product details based on matching pids. Then, it joins the merchants table to get the merchant details based on matching mids. In the results table, the query will select the product name and the merchant name as seller. The query is only looking for rows where quantity is zero, so that we get a list of names and sellers of products that are no longer available.

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

SQL QUERY

```
SELECT p.name, p.description
FROM products p
WHERE p.pid NOT IN (
       SELECT p.pid
       FROM products p
       INNER JOIN sell s ON p.pid = s.pid
);
```

SCREENSHOT



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

EXPLANATION

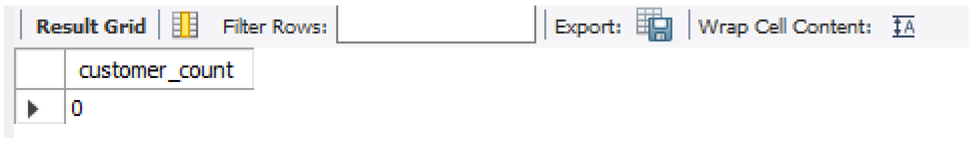
This query looks at products table. It has a subquery that will join the sell table to get the sell details based on matching pids to get all the pids that are being sold by some merchant. The parent query will then get pids that exist in the products table but does not exist in the subquery. In the results table, the query will select the product name and description so that we get a list of names and descriptions of products that are not sold.

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

SQL QUERY

```
WITH customers_with_SATA AS ( -- gets the cids for customers who bought a SATA drive
	SELECT DISTINCT c.cid
	FROM customers c
	INNER JOIN place p ON c.cid = p.cid
	INNER JOIN orders o ON p.oid = o.oid
	INNER JOIN contain ON o.oid = contain.oid
	WHERE contain.pid IN
		(SELECT pid FROM products WHERE name='Hard Drive' OR name='Super Drive')
), customers_with_routers AS ( -- gets the cids for customers who bought a router
	SELECT DISTINCT c.cid
	FROM customers c
	INNER JOIN place p ON c.cid = p.cid
	INNER JOIN orders o ON p.oid = o.oid
	INNER JOIN contain ON o.oid = contain.oid
	WHERE contain.pid IN
		(SELECT pid FROM products WHERE name='Router')
)
SELECT COUNT(*) AS customer_count FROM customers_with_SATA
WHERE cid NOT IN (SELECT cid FROM customers_with_routers);
```

SCREENSHOT



customer_count
0

EXPLANATION

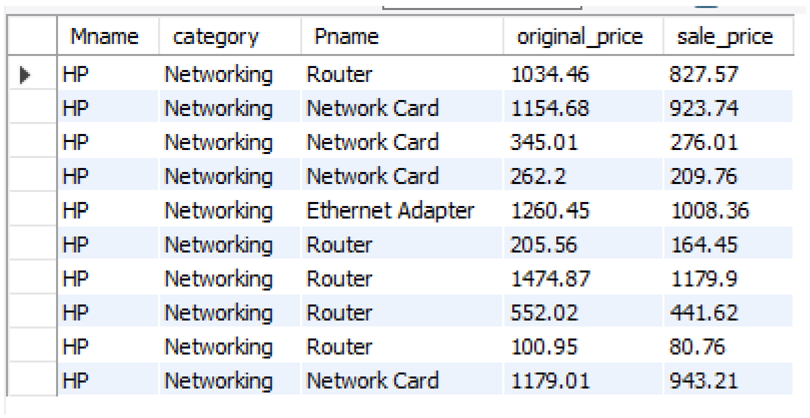
This query has two temporary relations: customers_with_SATA and customers_with_routers. customers_with_SATA looks at customers, place, orders, and contain tables. It joins place to get the place details based on matching customer IDs (cid), then joins orders to get the order details based on matching order IDs (oid), then lastly joins contain to get the contain details based on matching order IDs (oid). In the results table, the query will only select unique customer IDs to avoid duplicates. This query also has a subquery that looks at the products table to get product IDs for SATA drives. The CTE will then use this subquery to get the customer IDs who have bought SATA drives. customers_with _routers does the same, except in the subquery, it will get product IDs for routers, so that we get a list of customer IDs who have bought routers. The main query will then select the count of customer IDs that are in customers_with_SATA and not in customers_with_routers to get the number of customers who bought SATA drives but not any routers.

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

SQL QUERY

```
SELECT m.name AS Mname, p.category, p.name AS Pname, sell.price AS original_price,
ROUND(sell.price * 0.8, 2) AS sale_price
FROM merchants m
INNER JOIN sell ON m.mid = sell.mid
INNER JOIN products p ON sell.pid = p.pid
WHERE m.name='HP' AND p.category='Networking';
```

SCREENSHOT



Mname	category	Pname	original_price	sale_price
HP	Networking	Router	1034.46	827.57
HP	Networking	Network Card	1154.68	923.74
HP	Networking	Network Card	345.01	276.01
HP	Networking	Network Card	262.2	209.76
HP	Networking	Ethernet Adapter	1260.45	1008.36
HP	Networking	Router	205.56	164.45
HP	Networking	Router	1474.87	1179.9
HP	Networking	Router	552.02	441.62
HP	Networking	Router	100.95	80.76
HP	Networking	Network Card	1179.01	943.21

EXPLANATION

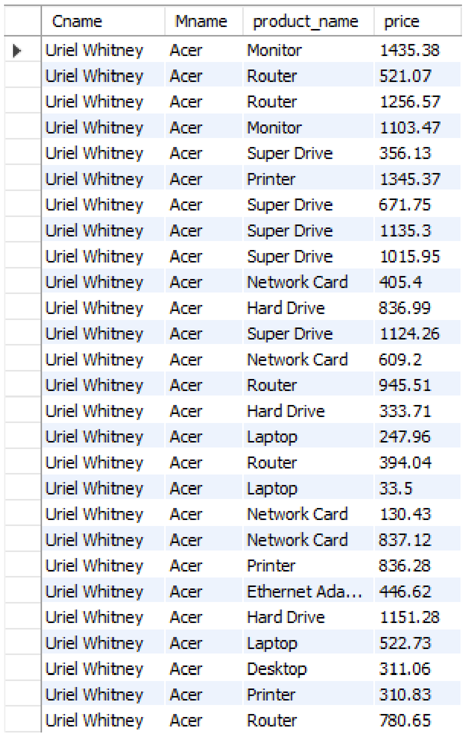
This query looks at merchants, sell, and products tables. It joins the sell table to get the sell details based on matching mids. Then, it joins the products table to get the products table to get the product details based on matching pids. In the results table, the query will select the merchant name, the product category, the product name, the original price, and 20% sale price. Then, the query will only display rows where the merchant name is HP and where the product category is Networking, so that we get a list of all HP Networking products.

5. What did Uriel Whitney order from Acer? (make sure to at least retrieve product names and prices).

SQL QUERY

```
SELECT c.fullname AS Cname, m.name AS Mname, p.name AS product_name, sell.price
FROM customers c
INNER JOIN place ON c.cid = place.cid
INNER JOIN contain ON place.oid = contain.oid
INNER JOIN products p ON contain.pid = p.pid
INNER JOIN sell ON p.pid = sell.pid
INNER JOIN merchants m ON sell.mid = m.mid
WHERE c.fullname='Uriel Whitney' AND m.name='Acer'
GROUP BY CName, Mname, product_name, sell.price;
```

SCREENSHOT



Cname	Mname	product_name	price
Uriel Whitney	Acer	Monitor	1435.38
Uriel Whitney	Acer	Router	521.07
Uriel Whitney	Acer	Router	1256.57
Uriel Whitney	Acer	Monitor	1103.47
Uriel Whitney	Acer	Super Drive	356.13
Uriel Whitney	Acer	Printer	1345.37
Uriel Whitney	Acer	Super Drive	671.75
Uriel Whitney	Acer	Super Drive	1135.3
Uriel Whitney	Acer	Super Drive	1015.95
Uriel Whitney	Acer	Network Card	405.4
Uriel Whitney	Acer	Hard Drive	836.99
Uriel Whitney	Acer	Super Drive	1124.26
Uriel Whitney	Acer	Network Card	609.2
Uriel Whitney	Acer	Router	945.51
Uriel Whitney	Acer	Hard Drive	333.71
Uriel Whitney	Acer	Laptop	247.96
Uriel Whitney	Acer	Router	394.04
Uriel Whitney	Acer	Laptop	33.5
Uriel Whitney	Acer	Network Card	130.43
Uriel Whitney	Acer	Network Card	837.12
Uriel Whitney	Acer	Printer	836.28
Uriel Whitney	Acer	Ethernet Ada...	446.62
Uriel Whitney	Acer	Hard Drive	1151.28
Uriel Whitney	Acer	Laptop	522.73
Uriel Whitney	Acer	Desktop	311.06
Uriel Whitney	Acer	Printer	310.83
Uriel Whitney	Acer	Router	780.65

EXPLANATION

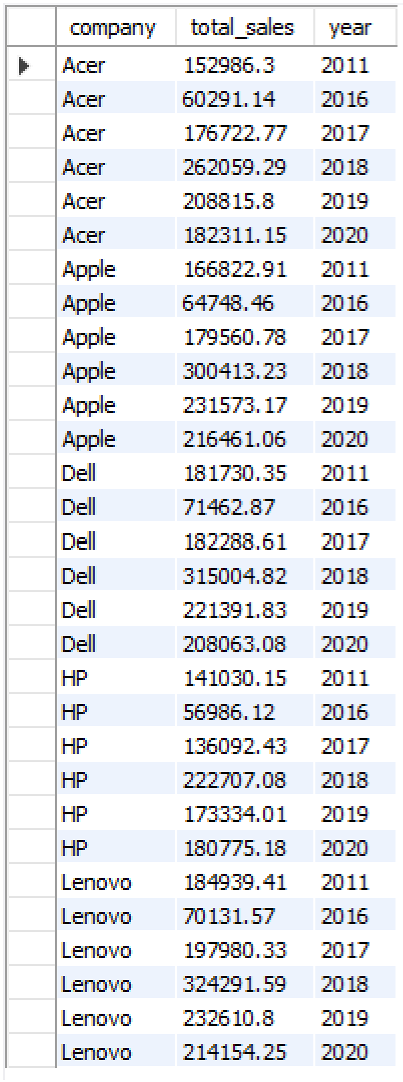
This query looks at customers, place, contain, products, sell, and merchants tables. It joins the place table to get the place details based on matching customer IDs (cid). Then, it joins the contain table to get the contain details based on matching order IDs (oid). Then, it joins the products table to get the product details based on matching product IDs (pid). Then, it joins the sell table to get the sell details based on matching product IDs (pid). Lastly, it joins the merchants table to get the merchant details based on matching merchant IDs (mid). In the results table, the query will select the customer name, the merchant name, the product name, and the product's price. The query will only look for rows where the customer's full name is 'Uriel Whitney' and where the merchant's name is 'Acer'. Then the results are grouped by customer name, merchant name, product name, and product price. This solution assumes that as long as Acer sells a given product, Uriel Whitney will buy that product from Acer, regardless of whether it is sold by other merchants. In order to fix this assumption, we would have to make changes to the relational model by adding a merchant id attribute to the contain table so that we can differentiate which merchant a customer purchased a product from. Alternatively, we can add an order_total attribute to the order table to use and compare against price in the sell table in order to differentiate which merchant a customer purchased a product from. Without either of these two modifications, we have no way of knowing which merchant a customer ordered from. Thus, I had to work off the assumption stated above.

6. List the annual total sales for each company (sort the results along the company and the year attributes).

SQL QUERY

```
SELECT m.name AS company, ROUND(SUM(sell.price), 2) AS total_sales,
YEAR(place.order_date) AS year
FROM merchants m
INNER JOIN sell ON m.mid = sell.mid
INNER JOIN contain ON sell.pid = contain.pid
INNER JOIN place ON contain.oid = place.oid
GROUP BY m.mid, year
ORDER BY company, year;
```

SCREENSHOT



company	total_sales	year
Acer	152986.3	2011
Acer	60291.14	2016
Acer	176722.77	2017
Acer	262059.29	2018
Acer	208815.8	2019
Acer	182311.15	2020
Apple	166822.91	2011
Apple	64748.46	2016
Apple	179560.78	2017
Apple	300413.23	2018
Apple	231573.17	2019
Apple	216461.06	2020
Dell	181730.35	2011
Dell	71462.87	2016
Dell	182288.61	2017
Dell	315004.82	2018
Dell	221391.83	2019
Dell	208063.08	2020
HP	141030.15	2011
HP	56986.12	2016
HP	136092.43	2017
HP	222707.08	2018
HP	173334.01	2019
HP	180775.18	2020
Lenovo	184939.41	2011
Lenovo	70131.57	2016
Lenovo	197980.33	2017
Lenovo	324291.59	2018
Lenovo	232610.8	2019
Lenovo	214154.25	2020

EXPLANATION

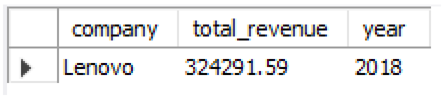
This query looks at merchants, sell, contain, and place tables. It first joins the sell table to get sell details based on matching merchant IDs (mid). Then, it joins the contain table to get contain details based on matching product IDs (pid). Lastly, it joins the place table to get place details based on matching order IDs (oid). In the results table, the query will select the merchant name with an alias company, the rounded sum of product prices with an alias total_sales, and the year. The results are grouped by merchant ID and year and then ordered by company and year, which gives us a list of the annual total sales for each company.

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

SQL QUERY

```
SELECT m.name AS company, ROUND(SUM(sell.price), 2) AS total_revenue,
YEAR(place.order_date) AS year
FROM merchants m
INNER JOIN sell ON m.mid = sell.mid
INNER JOIN contain ON sell.pid = contain.pid
INNER JOIN place ON contain.oid = place.oid
GROUP BY m.mid, year
ORDER BY total_revenue DESC
LIMIT 1;
```

SCREENSHOT



	company	total_revenue	year
▶	Lenovo	324291.59	2018

EXPLANATION

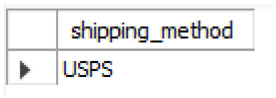
This query looks at merchants, sell, contain, and place tables. It joins the sell table to get the sell details based on matching merchant IDs (mid). Then, it joins the contain table to get the contain details based on matching product IDs (pid). Lastly, it joins the place table to get the place details based on matching order IDs (oid). In the results table, the query will display the merchant name with an alias company, the rounded sum of the prices with an alias total_revenue, and the year. The results are grouped by merchant ID (mid) and year and then ordered by total_revenue in descending sort. We limit the results by 1 so that we get the company with the highest annual revenue and the year with that revenue.

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

SQL QUERY

```
SELECT shipping_method
FROM orders
GROUP BY shipping_method
ORDER BY AVG(shipping_cost)
LIMIT 1;
```

SCREENSHOT



	shipping_method
▶	USPS

EXPLANATION

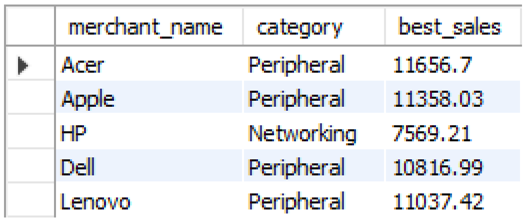
This query looks at orders table. In the results table, the query will select and group by shipping method then order by using an aggregate function, AVG, on shipping cost. Then, the query will limit the results by one so that we get the average cheapest shipping method used ever.

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

SQL QUERY

```
WITH category_sales AS ( -- gets total sales by category by merchant
    SELECT m.mid, m.name AS merchant_name, p.category, SUM(sell.price) AS total_sales
    FROM merchants m
    INNER JOIN sell ON m.mid = sell.mid
    INNER JOIN products p ON sell.pid = p.pid
    GROUP BY m.mid, m.name, p.category
),
max_category AS ( -- gets category with the highest sales by merchant
    SELECT mid, MAX(total_sales) AS max_sales
    FROM category_sales
    GROUP BY mid
)
SELECT sales.merchant_name, sales.category, ROUND(sales.total_sales, 2) AS best_sales
FROM category_sales sales
INNER JOIN max_category max ON sales.mid = max.mid AND sales.total_sales =
max.max_sales;
```

SCREENSHOT



merchant_name	category	best_sales
Acer	Peripheral	11656.7
Apple	Peripheral	11358.03
HP	Networking	7569.21
Dell	Peripheral	10816.99
Lenovo	Peripheral	11037.42

EXPLANATION

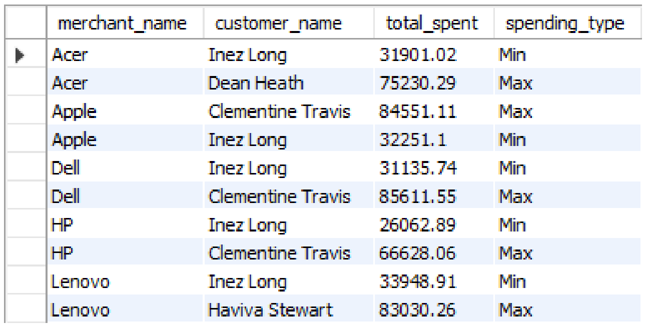
This query has two temporary relations: category_sales and max_category. category_sales looks at merchants, sell, and products tables. It joins sell to get the sell details based on matching merchant IDs (mid). Then, it joins products to get the product details based on matching product IDs (pid). In the results table, the query will select the merchant ID, the merchant name, the product category, and the sum of prices with an alias total_sales. The results are grouped by merchant ID, name, and product category, so that we get a CTE that gets the total sales by category by merchant. max_category then looks at category_sales and returns a relation with merchant IDs and the highest total sales from its best sold category. The main query looks at both of these CTEs and joins them together based on matching merchant IDs and sales, so that we get the list of best sold category for each company in a table with merchant name, category, and the total sales.

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

SQL QUERY

```sql
WITH customer_spending AS ( -- gets the list of how much each customer spent at each merchant
    SELECT m.mid, m.name AS merchant_name, c.cid, c.fullname AS customer_name,
       SUM(sell.price) AS total_spent
    FROM customers c
    INNER JOIN place ON c.cid = place.cid
    INNER JOIN contain ON place.oid = contain.oid
    INNER JOIN products p ON contain.pid = p.pid
    INNER JOIN sell ON p.pid = sell.pid
    INNER JOIN merchants m ON sell.mid = m.mid
    GROUP BY m.mid, m.name, c.cid, c.fullname
),
max_min_spent AS ( -- gets the max and min total spent by merchant
    SELECT mid, MAX(total_spent) AS max_spent, MIN(total_spent) AS min_spent
    FROM customer_spending
    GROUP BY mid
)
SELECT cs.merchant_name, cs.customer_name, ROUND(cs.total_spent, 2) AS total_spent,
       CASE WHEN cs.total_spent = mm.max_spent THEN 'Max'
            WHEN cs.total_spent = mm.min_spent THEN 'Min'
       END AS spending_type
FROM customer_spending cs
INNER JOIN max_min_spent mm ON cs.mid = mm.mid
WHERE cs.total_spent = mm.max_spent OR cs.total_spent = mm.min_spent
ORDER BY cs.merchant_name;
```

SCREENSHOT

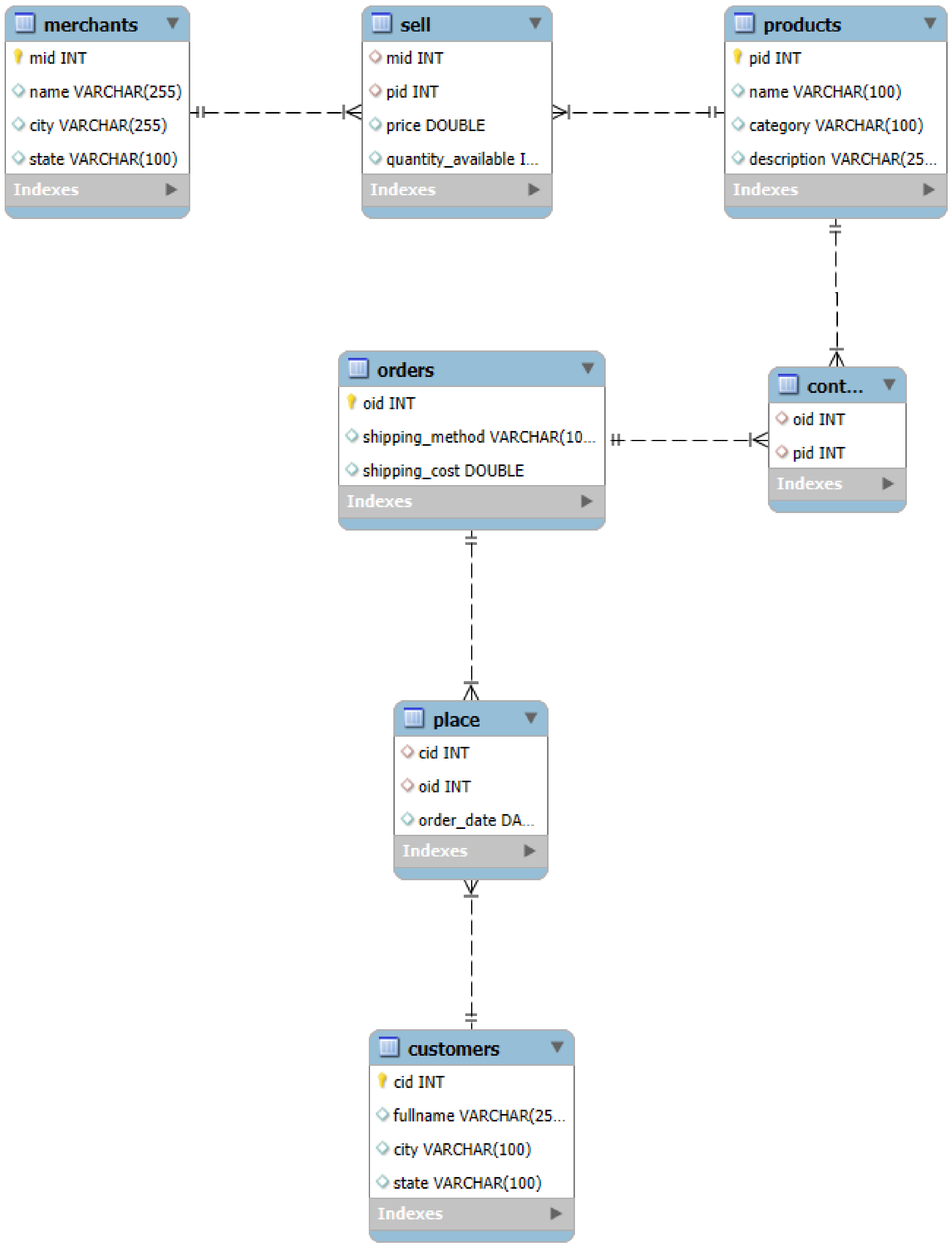


merchant_name	customer_name	total_spent	spending_type
Acer	Inez Long	31901.02	Min
Acer	Dean Heath	75230.29	Max
Apple	Clementine Travis	84551.11	Max
Apple	Inez Long	32251.1	Min
Dell	Inez Long	31135.74	Min
Dell	Clementine Travis	85611.55	Max
HP	Inez Long	26062.89	Min
HP	Clementine Travis	66628.06	Max
Lenovo	Inez Long	33948.91	Min
Lenovo	Haviva Stewart	83030.26	Max

EXPLANATION

This query has two temporary relations: customer_spending and max_min_spent. customer_spending looks at customers, place, contain, products, sell, and merchants tables. It first joins the place table based on matching customer IDs (cid). Then, it joins the contain table based on matching order IDs (oid). Then, it joins the products table based on matching product IDs (pid). Then, it joins the sell table based on matching product IDs (pid). Lastly, it joins the merchants table based on matching merchant IDs (mid). In the results table, the query selects the merchant ID, name, customer ID, name and the sum of price with an alias total_spent. The results are grouped by merchant ID, name, customer ID, and name to get a list of how much each customer spent at each merchant. max_min_spent looks at the prior CTE to get the max and min total spent from each merchant. The main query then joins these two CTEs and looks where total spent is equal to either the max or min spent. In the results table, the query will display the merchant name, customer name, the rounded total spent, and a spending type. The query uses a case expression to differentiate which is max and min spent. The results are then ordered by merchant name so that we get a list of customers who spent the least and most amount per company.

ERD Diagram



merchants
mid INT
name VARCHAR(255)
city VARCHAR(255)
state VARCHAR(100)
Indexes
sell
mid INT
pid INT
price DOUBLE
quantity_available I...
Indexes
products
pid INT
name VARCHAR(100)
category VARCHAR(100)
description VARCHAR(25...
Indexes
orders
oid INT
shipping_method VARCHAR(10...
shipping_cost DOUBLE
Indexes
cont...
oid INT
pid INT
Indexes
place
cid INT
oid INT
order_date DA...
Indexes
customers
cid INT
fullname VARCHAR(25...
city VARCHAR(100)
state VARCHAR(100)
Indexes