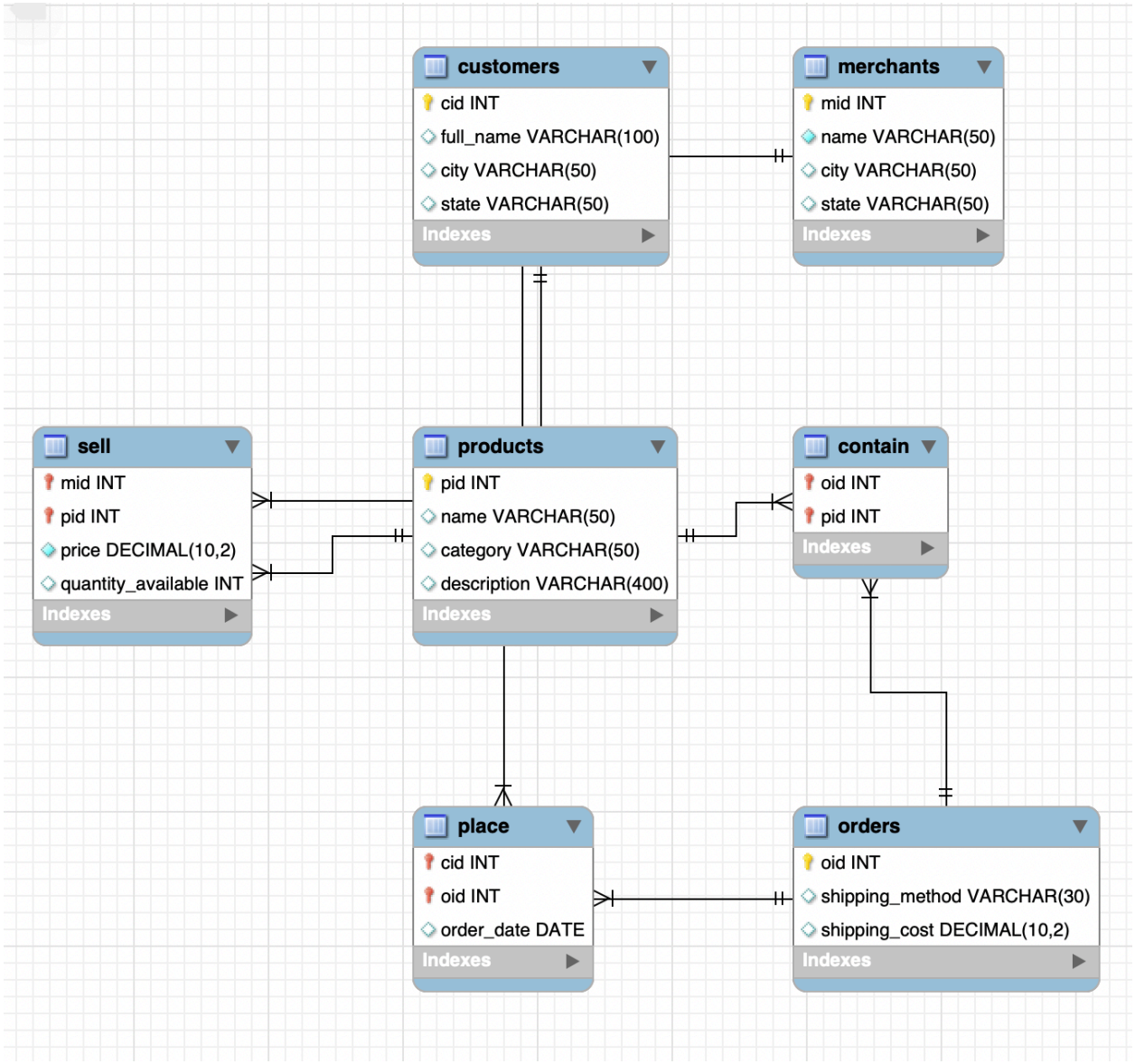
DB Assignment 3
Sarah Groark
5 October 2024

ERD:



Queries:

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

```sql
-- 1. names and sellers of products no longer available (quantity=0)
-- inner join: sell (mid), merchants (mid), products (pid)

select m.name, p.name
from sell s
inner join merchants m on s.mid = m.mid
inner join products p on s.pid = p.pid
where quantity_available=0;
```

This query retrieves the names of merchants and their corresponding products where the quantity available is at zero. The query starts with the sell table which contains records of which company sells certain products. Then, the sell table is joined to the merchants table on the *mid* to connect a sale of a product with the company name. The products table is also joined on *pid* to link this sale to the product name. The query outputs rows after filtering the results to display the records where quantity_available in the sell table is zero.
Output:

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

List names and descriptions of products that are not sold.

```sql
-- 2. list names and descriptions of products that aren't sold
-- left join: products (pid), sell (null pid)

select p.name as product_name, p.description
from products p left join sell s
    on (p.pid = s.pid)
where s.pid is null;
```

This query fetches the descriptions and names of products that are not sold by any merc. The query accomplishes this by joining the products table with the sell table on the *pid*. From there, the query filters the records to display rows only where there is no recorded corresponding *pid* found in the sell table.
Output:

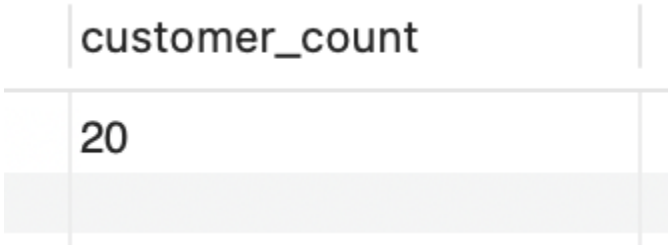
product_name	description
Super Drive	External CD/DVD/RW
Super Drive	UInternal CD/DVD/RW
Laptop	Intel Core i5-2410M 2.3GHz / 4GB RAM / 640G…
Laptop	Intel Core i5-2410M 2.3GHz / 4GB RAM / 640G…

How many customers bought SATA drives but not any routers?

```sql
-- 3. how many customers bought SATA drives but not any routers
-- inner join: place (oid), contain (oid), products (pid); filter by has name (SATA drives); filter by (no router)

select count(distinct place.cid) as customer_count
from place
inner join contain
    on place.oid = contain.oid
inner join products
    on contain.pid = products.pid
where products.description LIKE '%SATA%'
AND place.cid NOT IN (
    select distinct place.cid
    from place
    inner join contain
        on place.oid = contain.oid
    inner join products
        on contain.pid = products.pid
    where products.name = 'Router'
);
```

This query counts the number of distinct customers who have bought SATA drives, but *not* any routers. The query does this by joining the place table to the contain table (on *oid*) and the products table (on *pid*). The query then filters these records where the product name is equal to either “Hard Drive” or “Super Drive”. The subquery then performs the same joins but filters by “Router” instead. The subquery, using keywords ‘NOT IN’ ensures that these customers who also bought routers are excluded from the main query’s results.
Output:



customer_count
20

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

```
-- 4. HP has a 20% sale on all of its networking products
-- update sell table (price) on pid and mid where merchant = HP and product category = networking

UPDATE sell s
INNER JOIN products p ON s.pid = p.pid
INNER JOIN merchants m ON s.mid = m.mid
SET s.price = s.price * 0.8  -- Apply a 20% discount
WHERE p.category = 'Networking' AND m.name = 'HP';
```

This query updates the sell table, specifically on the price column. The specific rows that are changed are a result of the two inner join statements: sell is joined on the products table on *pid* and as well as the merchants table on *mid*. The query then updates the rows of the sell table where the merchant corresponds to HP and the corresponding product category is networking.
Output: shows prices before and after the sale

company	product	original	sale_price
HP	Router	1034.46	827.57
HP	Network Card	1154.68	923.74
HP	Network Card	345.01	276.01
HP	Network Card	262.20	209.76
HP	Ethernet Adapter	1260.45	1008.36
HP	Router	205.56	164.45
HP	Router	1474.87	1179.90
HP	Router	552.02	441.62
HP	Router	100.95	80.76
HP	Network Card	1179.01	943.21

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

```
-- 5. What did Uriel Whitney order from Acer?
-- inner join: customers (cid), contain (oid), products (pid), sell (pid), merchants (mid)

select merchants.name as company, customers.full_name as customer_name, products.name as product, sell.price
from customers
inner join place on (customers.cid = place.cid)
inner join contain on (place.oid = contain.oid)
inner join products on (contain.pid = products.pid)
inner join sell on (products.pid = sell.pid)
inner join merchants on (sell.mid = merchants.mid)
where merchants.name = 'Acer' and customers.full_name = 'Uriel Whitney'; -- filter by what Uriel Whitney bought only from Acer
```

This query joins the customers table on the place table (on *cid*), the contain table (on *oid*), products (*pid*), sell (*pid*), and merchants (*mid)*. This outputs rows to pull information about what customers have bought from certain merchants. The query then filters these rows to only show records where the customer name is Uriel Whitney and the merchant name is Acer. As a result, the records shown represent what Uriel Whitney has purchased from Acer.

Output:

company	customer_name	product	price
Acer	Uriel Whitney	Super Drive	356.13
Acer	Uriel Whitney	Network Card	130.43
Acer	Uriel Whitney	Hard Drive	836.99
Acer	Uriel Whitney	Printer	310.83
Acer	Uriel Whitney	Printer	1345.37
Acer	Uriel Whitney	Super Drive	356.13
Acer	Uriel Whitney	Super Drive	1015.95
Acer	Uriel Whitney	Network Card	405.40
Acer	Uriel Whitney	Monitor	1103.47
Acer	Uriel Whitney	Super Drive	1135.30
Acer	Uriel Whitney	Router	1256.57
Acer	Uriel Whitney	Super Drive	356.13
Acer	Uriel Whitney	Printer	1345.37
Acer	Uriel Whitney	Super Drive	671.75
Acer	Uriel Whitney	Router	394.04
Acer	Uriel Whitney	Laptop	33.50
Acer	Uriel Whitney	Laptop	247.96
Acer	Uriel Whitney	Router	521.07
Acer	Uriel Whitney	Hard Drive	836.99
Acer	Uriel Whitney	Super Drive	1135.30
Acer	Uriel Whitney	Super Drive	1124.26
Acer	Uriel Whitney	Network Card	609.20
Acer	Uriel Whitney	Printer	1345.37
Acer	Uriel Whitney	Network Card	405.40
Acer	Uriel Whitney	Super Drive	671.75
Acer	Uriel Whitney	Router	521.07
Acer	Uriel Whitney	Super Drive	1015.95
Acer	Uriel Whitney	Super Drive	671.75
Acer	Uriel Whitney	Laptop	247.96
Acer	Uriel Whitney	Router	1256.57
Acer	Uriel Whitney	Network Card	130.43
Acer	Uriel Whitney	Monitor	1103.47
Acer	Uriel Whitney	Laptop	33.50
Acer	Uriel Whitney	Network Card	837.12
Acer	Uriel Whitney	Hard Drive	1151.28
Acer	Uriel Whitney	Laptop	247.96
Acer	Uriel Whitney	Hard Drive	836.99
Acer	Uriel Whitney	Monitor	1103.47
Acer	Uriel Whitney	Printer	836.28
Acer	Uriel Whitney	Router	1256.57
Acer	Uriel Whitney	Super Drive	1015.95
Acer	Uriel Whitney	Router	394.04
Acer	Uriel Whitney	Hard Drive	333.71
Acer	Uriel Whitney	Laptop	247.96
Acer	Uriel Whitney	Router	945.51
Acer	Uriel Whitney	Network Card	837.12
Acer	Uriel Whitney	Printer	836.28
Acer	Uriel Whitney	Ethernet Ad…	446.62
Acer	Uriel Whitney	Super Drive	1015.95
Acer	Uriel Whitney	Network Card	405.40
Acer	Uriel Whitney	Hard Drive	1151.28
Acer	Uriel Whitney	Printer	836.28
Acer	Uriel Whitney	Laptop	522.73

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

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

select merchants.name as company_name, year(place.order_date) as year, sum(sell.price) as annual_sales
from place
inner join contain on (place.oid = contain.oid)
inner join sell on (contain.pid = sell.pid)
inner join merchants on (sell.mid = merchants.mid)
group by merchants.name, year
order by year;
```

This query retrieves the sum of all sales made by each company by year. The place table (which tracks the orders placed) joins with the contain table (by oid), the sell table (pid), and merchants table (by mid). The query then groups the results by company and by year, so the output represents the annual sales of each company by year.

Output:

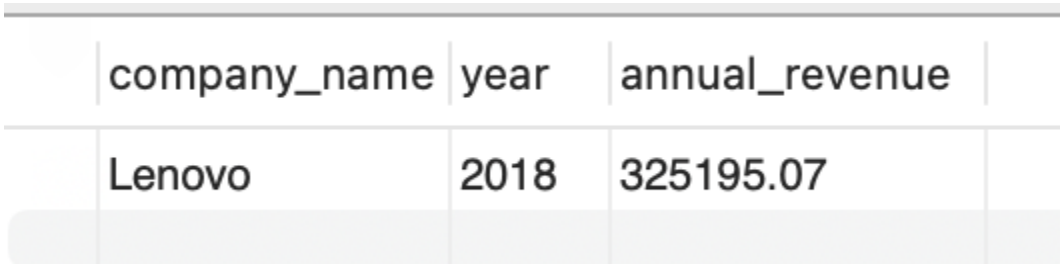
company_name	year	annual_sales
Acer	2011	152986.30
Apple	2011	166822.91
HP	2011	128208.36
Dell	2011	181730.35
Lenovo	2011	184939.41
Acer	2016	60291.14
Apple	2016	64748.46
HP	2016	52167.53
Dell	2016	71462.87
Lenovo	2016	70131.57
Acer	2017	176722.77
Apple	2017	179560.78
HP	2017	124796.74
Dell	2017	182288.61
Lenovo	2017	197980.33
Acer	2018	262896.28
Apple	2018	301741.42
HP	2018	202028.84
Dell	2018	315975.27
Lenovo	2018	325195.07
Acer	2019	208815.80
Apple	2019	231573.17
HP	2019	156175.84
Dell	2019	221391.83
Lenovo	2019	232610.80
Acer	2020	182311.15
Apple	2020	216461.06
HP	2020	164084.33
Dell	2020	208063.08
Lenovo	2020	214154.25

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

```
-- 7. which company had the highest annual revenue and in what year?
-- inner join: place (oid), contain (oid), sell (pid), merchants (mid), group by company & year, order limit by top 1


select merchants.name as company_name, year(place.order_date) as year, sum(sell.price) as annual_revenue
from place
inner join contain on (place.oid = contain.oid)
inner join sell on (contain.pid = sell.pid)
inner join merchants on (sell.mid = merchants.mid)
group by merchants.name, year
order by annual_revenue desc
limit 1;
```

This query performs the same process found in query number 6, but limits the output to the highest annual_revenue. The output still reports the company name, the annual revenue, and the associated year of that revenue.
Output:



company_name	year	annual_revenue
Lenovo	2018	325195.07

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

```
-- 8. on average, what was the cheapest shipping method used ever
-- reports cheapest shipping cost of all orders (on average); uses orders table to filter output

select orders.shipping_method, round(avg(orders.shipping_cost),2) as average_price
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
)
order by avg(orders.shipping_cost);
```

This query utilizes only one table, the orders table, to take the average of the shipping costs by shipping method. Then, the lowest average is returned by using the having clause, to ensure the true cheapest shipping method is returned.
Output:

shipping_method	average_price
USPS	7.46

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

```
-- 9. what is the best sold ($) category for each company?
-- name (merchants), category (products), sell (price), dontain (pid)

WITH CategorySales AS (

    select m.name as merchant_name, p.category, sum(s.price) as total_sales
    from products p
    inner join contain c on (p.pid = c.pid)
    inner join sell s on (p.pid = s.pid)
    inner join merchants m on (s.mid = m.mid)
    group by m.name, p.category
),
MaxSales AS (

    select merchant_name, max(total_sales) as max_sales
    from CategorySales
    group by merchant_name

)
select cs.merchant_name, cs.category, cs.total_sales
from CategorySales cs
inner join MaxSales ms on (cs.merchant_name = ms.merchant_name and cs.total_sales = ms.max_sales);
```

This query uses two CTEs. The first one, CategorySales, calculates the sales totals of all categories sold by each company; the CTE organizes this calculation by the products table, contain table, sell table, and merchants table. The second, MaxSales, organizes these sales by selecting the maximum sales total for each company. Then, the main query joins these two CTEs to display the best selling category for each of the 5 companies.
Output:

merchant_name	category	total_sales
Acer	Peripheral	752542.65
Apple	Peripheral	726729.63
HP	Peripheral	416673.29
Dell	Peripheral	691296.94
Lenovo	Peripheral	703695.42

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

```
-- 10. for each company find out which customer has spent the least and most amounts
-- place (oid), contain (oid -> pid), sell (price), merchants (mid)

WITH CustomerSpending AS (
    select m.name as company, c.full_name, sum(s.price) as total_spent
    from customers c
    inner join place p on (c.cid = p.cid)
    inner join contain con on (p.oid = con.oid)
    inner join sell s on (con.pid = s.pid)
    inner join merchants m on (s.mid = m.mid)
    group by m.name, c.full_name

),
HighestSpender AS(

    select cs.company, cs.full_name, cs.total_spent
    from CustomerSpending cs
    where total_spent = (
        select max(total_spent)
        from CustomerSpending cs2
        where cs2.company = cs.company
    )
),
LowestSpender AS(
    select cs.company, cs.full_name, cs.total_spent
    from CustomerSpending cs
    where total_spent = (
        select min(total_spent)
        from CustomerSpending cs3
        where cs3.company = cs.company

    )
)

select h.company, h.full_name as highest_spender, h.total_spent as highest_spent, l.full_name as lowest_spender, l.total_spent as lowest_spent
from HighestSpender h
inner join LowestSpender l on (h.company = l.company);
```

This query also uses CTEs. The first, CustomerSpending, calculates the spending of each customer at each company. HighestSpender then, for each company, determines the customer who has spent the most money. LowestSpender, similarly, determines the customer that has spent the least amount of money at each of the 5 companies. Finally, in the main query, the HighestSpender and LowestSpender CTEs are joined to display the following for each of the five

companies: highest spending customer name, spent amount for that customer, lowest spending customer name, and the lowest spent amount for that customer.
Output:

company	highest_spender	highest_spent	lowest_spender	lowest_spent
Acer	Dean Heath	75230.29	Inez Long	31901.02
Apple	Clementine Travis	84551.11	Inez Long	32251.10
HP	Clementine Travis	60203.35	Inez Long	23483.83
Dell	Clementine Travis	85611.55	Inez Long	31135.74
Lenovo	Haviva Stewart	83030.26	Inez Long	33948.91