Title Page

- **Title:** DB Assignment 3
- **Name**: Nicolas Zywalewski
- **Due Date:** 21 October 2025

SQL Section

- Creating constraints along with primary/foreign keys, consistent with the assignment directions:

```
alter table merchants
add constraint mer_pk primary key(mid);

alter table products
add constraint prod_pk primary key(pid),
add constraint prod_name_const check
    (name in ("Printer", "Ethernet Adapter", "Desktop", "Hard Drive",
    "Laptop", "Router", "Network Card", "Super Drive", "Monitor")),
add constraint prod_cat_const check (category in
    ("Peripheral", "Networking", "Computer"));

alter table sell
add constraint sell_fk_mid foreign key(mid)
    references merchants(mid)
    on delete cascade
    on update cascade,
add constraint sell_fk_pid foreign key(pid)
    references products(pid)
    on delete cascade
    on update cascade,
add constraint sell_price_const check
    (price between 0 and 100000),
add constraint sell_qa_const check
    (quantity_available between 0 and 1000);
```

```
alter table orders
add constraint orders_pk primary key(oid),
add constraint orders_sm_const check
    (shipping_method in ("UPS","FedEx","USPS")),
add constraint orders_sc_const check
    (shipping_cost between 0 and 500);

alter table contain
add constraint con_fk_oid foreign key(oid)
    references orders(oid)
    on delete cascade
    on update cascade,
add constraint con_fk_pid foreign key(pid)
    references products(pid)
    on delete cascade
    on update cascade;

alter table customers
add constraint cust_pk primary key(cid);

alter table place
add constraint place_fk_cid foreign key(cid)
    references customers(cid)
    on delete cascade
    on update cascade,
add constraint place_fk_oid foreign key(oid)
    references orders(oid)
    on delete cascade
    on update cascade;
```

Problem 1:

```
62    -- Query 1: List names and sellers of products that are no longer available (quantity=0)
63    -- Using merchants, products, and sell tables
64 •  select m.name as Seller,
65        p.name as ProductName,
66        s.quantity_available as Quantity
67    from merchants m
68    inner join sell s on m.mid = s.mid
69    inner join products p on s.pid = p.pid
70    where s.quantity_available = 0;
```

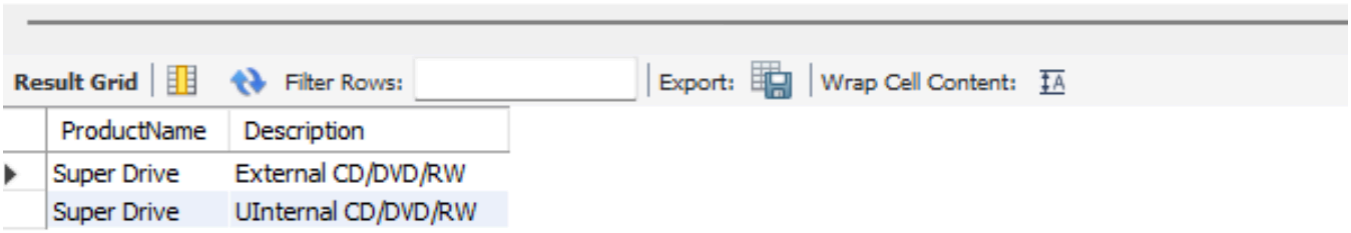
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Seller	ProductName	Quantity
Acer	Router	0
Acer	Network Card	0
Apple	Printer	0
Apple	Router	0
HP	Router	0
HP	Super Drive	0
HP	Laptop	0
Dell	Router	0
Lenovo	Ethernet Adapter	0

Explanation: The merchants, sell, and products tables are joined on mid and pid. I filtered rows where sell.quantity_available = 0 to find out of stock items. The output returns seller name, product name, and the current quantity.

Problem 2:

```
72      -- Query 2: List names and descriptions of products that are not sold.
73      -- Left join with products table on left, so all products are included
74      select p.name as ProductName, p.description as Description
75      from products p
76      left outer join sell s
77          on s.pid = p.pid
78      where s.pid is NULL; -- filtering for only the products that are not sold
79
```

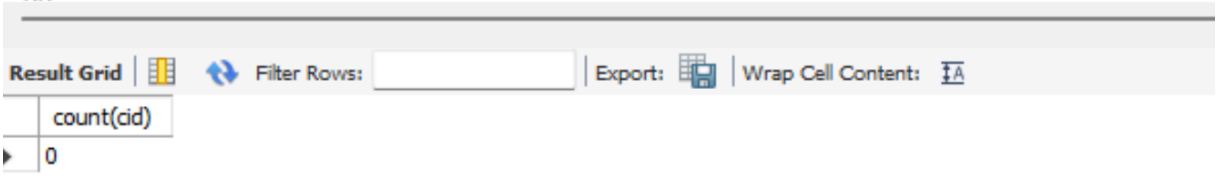


ProductName	Description
Super Drive	External CD/DVD/RW
Super Drive	UInternal CD/DVD/RW

Explanation: The products table is left-joined with sell so that all products are included. Rows where the ProductID from sell is NULL are kept, showing only products not currently sold. The output returns product name and description.

Problem 3:

```
80      -- Query 3: How many customers bought SATA drives but not any routers?
81      select count(cid) from
82          ( -- part one gets all customers that ordered a Hard Drive
83          select distinct cust.cid
84          from customers cust
85          inner join place on place.cid = cust.cid
86          inner join contain con on con.oid = place.oid
87          inner join products prod on prod.pid = con.pid
88          where prod.name = 'Hard Drive'
89
90          except  -- we are taking away the customers that also bought a Router
91
92          select distinct cust.cid
93          from customers cust
94          inner join place on place.cid = cust.cid
95          inner join contain con on con.oid = place.oid
96          inner join products prod on prod.pid = con.pid
97          where prod.name = 'Router'
98          ) t;
```



Explanation: The customers, place, contain, and products tables are joined to find all customers who bought Hard Drives. A similar join is used to find customers who bought Routers, Using “except”, router buyers are removed, and the remaining count of customers is returned.
Note: I interpreted SATA drives to be Hard Drives, based upon what I found from a Google Search, and a lack of other potential matches that I could see within our data.

Problem 4:

```
101     -- Query 4: HP has a 20% sale on all its Networking products.
102     -- Interpreted this as asking to show the old and prices of each product that changed
103 •   select m.name as Seller,
104         p.name as ProductName,
105         p.category as Category,
106         s.price as OldPrice,
107         (s.price * 0.80) as NewPrice
108     from merchants m
109     inner join sell s
110         on m.mid = s.mid
111     inner join products p
112         on p.pid = s.pid
113     where m.name = 'HP' and p.category = 'Networking';
114
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Seller	ProductName	Category	OldPrice	NewPrice
HP	Router	Networking	1034.46	827.5680000000001
HP	Network Card	Networking	1154.68	923.7440000000001
HP	Network Card	Networking	345.01	276.008
HP	Network Card	Networking	262.2	209.76
HP	Ethernet Adapter	Networking	1260.45	1008.3600000000001
HP	Router	Networking	205.56	164.448
HP	Router	Networking	1474.87	1179.896
HP	Router	Networking	552.02	441.616
HP	Router	Networking	100.95	80.76
HP	Network Card	Networking	1179.01	943.2080000000001

Explanation: The merchants, sell, and products tables are joined together. Filtering on seller ‘HP’ and category “Networking” finds the relevant products. A discounted price is calculated by multiplying the old price by 0.8, and both prices are returned.

Problem 5:

```
115     -- Query 5: What did Uriel Whitney order from Acer? (make sure to at least retrieve product names and prices).
116     -- This query is a bit difficulty because our schema does not allow us to see which merchant supplied the product that Uriel Whitney purchased.
117     -- Each product can be sold by multiple merchants, so we have no way of knowing which of the products that Uriel Whitney bought were from Acer.
118     -- This query answers the question: What items that Uriel Whitney ordered does Acer sell?
119         -- This is as close as we can get with the current schema.
120     select distinct p.pid as ProductID, p.name as ProductName, s.price as Price, m.name as Seller
121     from customers cust
122     inner join place on place.cid = cust.cid
123     inner join contain c on c.oid = place.oid
124     inner join products p on p.pid = c.pid
125     inner join sell s on s.pid = p.pid
126     inner join merchants m on m.mid = s.mid
127     where cust.fullname = 'Uriel Whitney' and m.name = 'Acer'
128     order by p.pid;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ProductID	ProductName	Price	Seller
1	Hard Drive	836.99	Acer
2	Monitor	1103.47	Acer
3	Printer	310.83	Acer
5	Hard Drive	333.71	Acer
6	Laptop	247.96	Acer
7	Super Drive	1135.3	Acer
8	Router	521.07	Acer
10	Network Card	837.12	Acer
11	Super Drive	1124.26	Acer
13	Network Card	609.2	Acer
14	Monitor	1435.38	Acer
15	Printer	836.28	Acer
16	Ethernet Ad...	446.62	Acer
17	Desktop	311.06	Acer
18	Router	1256.57	Acer
19	Router	780.65	Acer
20	Router	945.51	Acer
21	Super Drive	356.13	Acer
22	Super Drive	1015.95	Acer
23	Router	394.04	Acer
24	Laptop	522.73	Acer
25	Laptop	33.5	Acer
26	Printer	1345.37	Acer
27	Network Card	405.4	Acer
28	Network Card	130.43	Acer
29	Hard Drive	1151.28	Acer
30	Super Drive	671.75	Acer

Result 44

Explanation: The customers, place, contain, products, sell, and merchants tables are all joined together to answer this query. Filtering on customer “Uriel Whitney” and seller “Acer” returns products Uriel ordered that Acer also sells. Because the schema does not record which merchant actually supplied an order, this result only shows an overlap rather than confirmed purchases from Acer.

Problem 6:

```
135     -- Query 6: List the annual total sales for each company (sort the results along the company and the year attributes).
136     -- The current schema does not allow us to answer this query specifically.
137         -- This is because the current schema does not allow us to see which merchant supplied each product that was bought.
138         -- The orders table tells us which products were purchased in each order, but not which merchant supplied the product.
139     -- The following query inflates totals when multiple merchants sell the same product, because `contain` does not store `mid`.
140         -- Results should be interpreted as potential revenue, not actual sales.
141 •   select m.name as Company,
142         year(pl.order_date) as OrderYear,
143         sum(s.price*s.quantity_available) as PotentialRev
144     from sell s
145     inner join contain c
146         on c.pid = s.pid
147     inner join place pl
148         on pl.oid = c.oid
149     inner join merchants m
150         on m.mid = s.mid
151     group by Company, OrderYear
152     order by OrderYear, Company;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Company	OrderYear	PotentialRev
Acer	2011	828677.0800000004
Apple	2011	972240.9199999999
Dell	2011	1542228.989999998
HP	2011	873547.1000000003
Lenovo	2011	1235551.84
Acer	2016	307909.83
Apple	2016	409402.38
Dell	2016	625684.1399999997
HP	2016	375547.4499999999
Lenovo	2016	483906.56
Acer	2017	1100206.8500000006
Apple	2017	1071712.9300000002
Dell	2017	1522794.2700000084

Result 2

Explanation: The sell, contain, place, and merchants tables are joined together. Grouping by company and order year, the query sums price*quantity_available to estimate revenues. The schema does not link merchants to specific orders, so totals are inflated and represent potential revenues, not actual sales.

Problem 7:

```
154     -- Query 7: Which company had the highest annual revenue and in what year?
155     -- The current schema does not allow us to answer this query specifically.
156         -- This is because the current schema does not allow us to see which merchant supplied each product that was bought.
157     -- The following query inflates totals when multiple merchants sell the same product, because `contain` does not store `mid`.
158         -- Results should be interpreted as potential revenue, not actual sales.
159 •   select Company, OrderYear, PotentialRev as HighestRev
160     from (
161         select m.name as Company, year(pl.order_date) as OrderYear, sum(s.price*s.quantity_available) as PotentialRev
162         from sell s
163         inner join contain c on c.pid = s.pid
164         inner join place pl on pl.oid = c.oid
165         inner join merchants m on m.mid = s.mid
166         group by Company, OrderYear
167         order by OrderYear, Company
168     ) t
169     order by PotentialRev
170     limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Company	OrderYear	HighestRev
Acer	2016	307909.83

Explanation: The same joins as Query 6 are used to calculate yearly totals per company. The result is ordered by revenue and the top row is selected as the highest. The totals are inflated for the same reason as Query 6, and they represent potential revenue, not actual sales.

Problem 8:

```
173     -- Query 8: On average, what was the cheapest shipping method used ever?
174     -- This query can be answered directly, by calculating average shipping cost, grouped by the method
175     -- The average shipping costs were sorted from lowest to highest, and only the lowest is shown with a limit of 1 being used.
176 •   select shipping_method as ShippingMethod, avg(shipping_cost) as AvgCost
177     from orders
178     group by shipping_method
179     order by AvgCost
180     limit 1;
181
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

ShippingMethod	AvgCost
USPS	7.455760869565214

Explanation: The orders table is grouped by shipping method. The average shipping cost is calculated for each method, and the cheapest one is returned by ordering and limiting the results.

Problem 9:

```sql
-- Query 9: What is the best sold ($) category for each company?
-- The current schema does not allow us to answer this query specifically.
    -- This is because the schema does not record which merchant actually fulfilled each purchased product.
    -- In addition, the schema does not record quantities sold, only quantities available for sale.
-- The following query therefore uses inventory data (price × quantity_available from `sell`) to approximate revenue.
    -- Results should be interpreted as potential revenue, not actual sales.
select t1.Company, t1.Category, t1.PotentialRev from
(
    -- this subquery gets the potential revenue for each category for each company
    select m.name as Company, p.category as Category, sum(s.price*s.quantity_available) as PotentialRev
    from sell s
    inner join merchants m on m.mid = s.mid
    inner join products p on p.pid = s.pid
    group by m.name, p.category
) as t1
where t1.PotentialRev = (select max(t2.PotentialRev) -- we select the highest potential revenue for each company
    from (
        -- this subquery also gets the potential revenue for each category for each company
        select m2.name as Company, p2.category as Category, sum(s2.price*s2.quantity_available) as PotentialRev
        from sell s2
        inner join merchants m2 on m2.mid = s2.mid
        inner join products p2 on p2.pid = s2.pid
        group by m2.name, p2.category
    ) as t2
    where t1.Company = t2.Company -- we select the highest potential revenue for each company
) order by t1.Company; -- we put the companies in alphabetical order, for display
```

Company	Category	PotentialRev
Acer	Peripheral	78136.53
Apple	Peripheral	63974.73999999999
Dell	Peripheral	100753.96000000002
HP	Peripheral	51133.47
Lenovo	Peripheral	83479.82999999999

Explanation: The merchants sell, and products tables are joined to answer this query. For each company, category totals are calculated using price*quantity_available, and the maximum category per company is selected. The output shows the category with the highest potential revenue, but results are based on inventory, not actual sales.

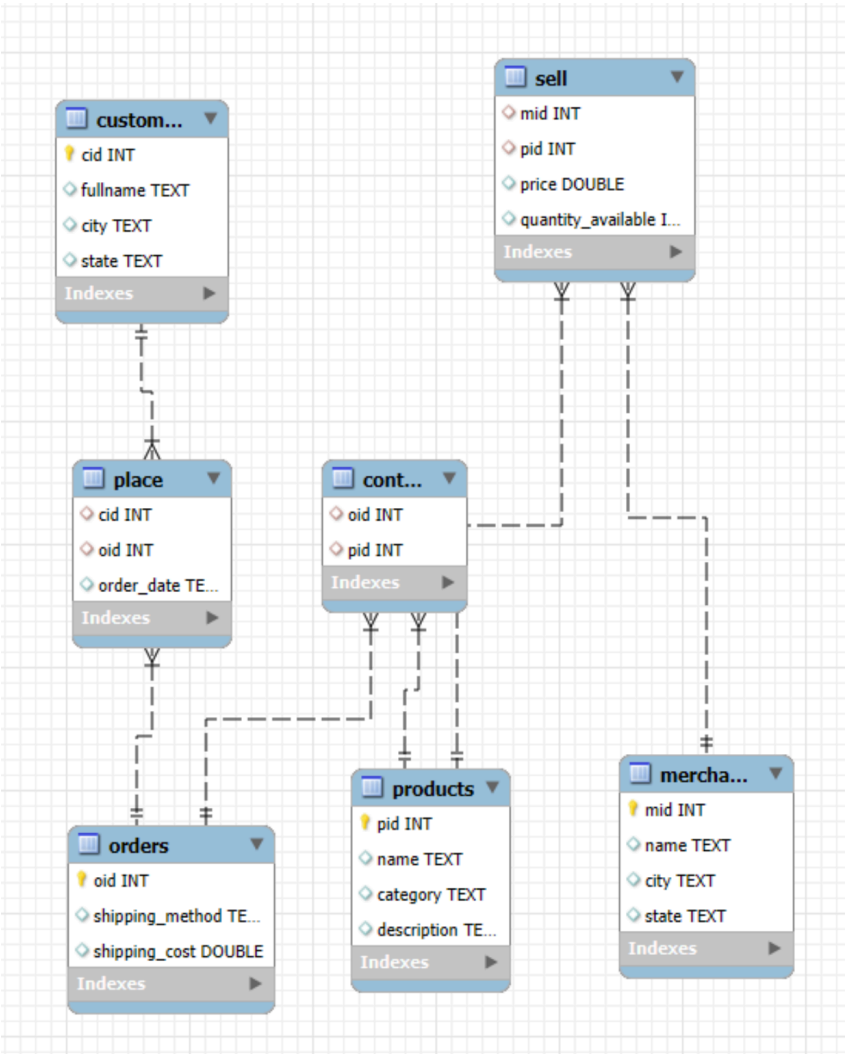
Problem 10:

```sql
-- Query 10: For each company, find out which customers have spent the most and the least amounts.
-- The current schema does not allow us to answer this query specifically.
    -- This is because `contain` does not store mid, so we cannot tell which merchant actually fulfilled the product.
-- The query assumes qty = 1 and uses prices from `sell`.
-- Totals are inflated when multiple merchants sell the same product, since each merchant is credited.
-- Results should be read as potential spend, not actual sales.
with customers_per_company as (
    -- this query builds a table of how much each customer spends with each company
    select m.name Company, c.fullname CustomerName, sum(s.price) as Spent
    from customers c
    inner join place p on p.cid = c.cid
    inner join contain co on co.oid = p.oid
    inner join sell s on s.pid = co.pid
    inner join merchants m on m.mid = s.mid
    group by m.name, c.fullname)
-- for each company, select the customers that spent the most
select t1.Company, t1.CustomerName, t1.Spent
from customers_per_company t1
where t1.Spent = (
    select max(t2.Spent) from customers_per_company t2
    where t2.Company = t1.Company)
union all -- union to combine results
-- for each company, select the customers that spent the least
select t1.Company, t1.CustomerName, t1.Spent
from customers_per_company t1
where t1.Spent = (
    select min(t2.Spent) from customers_per_company t2
    where t2.Company = t1.Company)
order by company;
```

Company	CustomerName	Spent
Acer	Dean Heath	75230.28999999998
Acer	Inez Long	31901.019999999993
Apple	Clementine Travis	84551.10999999997
Apple	Inez Long	32251.099999999988
Dell	Clementine Travis	85611.54999999999
Dell	Inez Long	31135.74000000001
HP	Clementine Travis	66628.05999999995
HP	Inez Long	26062.89
Lenovo	Haviva Stewart	83030.25999999997
Lenovo	Inez Long	33948.909999999996

Explanation: The customers, place, contain, sell, and merchants tables are joined to build customer totals for each company. A CTE is used to calculate spending, and the maximum and minimum spenders per company are returned using a Union. Because the schema does not record which merchant supplied an order, totals are inflated and represent potential spend rather than actual sales.

ERD Diagram:


custom...
cid INT
fullname TEXT
city TEXT
state TEXT
Indexes
sell
mid INT
pid INT
price DOUBLE
quantity_available I...
Indexes
place
cid INT
oid INT
order_date TE...
Indexes
cont...
oid INT
pid INT
Indexes
orders
oid INT
shipping_method TE...
shipping_cost DOUBLE
Indexes
products
pid INT
name TEXT
category TEXT
description TE...
Indexes
mercha...
mid INT
name TEXT
city TEXT
state TEXT
Indexes