Database Lab Report 5

Course: Database Management
Lab Number: *Lab #5
Date: *2025-02-18*
Name: *Ryan Munger*

1. Objective

Write some very interesting SQL queries and work a little harder for the lab points.

2. Lab Setup

A fresh CAP database and knowledge of table joins & subqueries.

3. Procedure

Part 1: *Queries*

For each question, write the query. Then, use AI to generate the queries and grade its response.

1. Show all the People data (and only people data) for people who are customers. Use joins this time; no subqueries.

```sql
-- Me
SELECT p.*
FROM People p INNER JOIN Customers c on p.pid = c.pid;
-- AI
SELECT p.*
FROM People p
JOIN Customers c ON p.pid = c.pid;
```

Grade: 9/10, don't assume reader knows the default join!

2. Show all the People data (and only the people data) for people who are agents. Use joins this time; no subqueries.

```sql
-- Me
SELECT p.*
FROM People p INNER JOIN Agents a on p.pid = a.pid;
-- AI
SELECT p.*
FROM People p
JOIN Agents a ON p.pid = a.pid;
```

Grade: 9/10, defaulting again I see

3. Show all People and Agent data for people who are both customers and agents. Use joins this time; no subqueries.

```
-- Me
SELECT p.*, a.*
FROM People p INNER JOIN Agents a on p.pid    = a.pid
              INNER JOIN Customers c on p.pid = c.pid;
-- AI
SELECT p.*, a.*
FROM People p
JOIN Customers c ON p.pid = c.pid
JOIN Agents a ON p.pid = a.pid;
```

Grade: 9/10; I see a pattern here

4. Show the first name of customers who have never placed an order. Use subqueries.

```
-- Me
SELECT firstName
FROM People
WHERE pid IN (
              SELECT pid
              FROM Customers
              WHERE pid NOT IN (
                                SELECT custId
                                FROM Orders
                               )
             );
-- AI
SELECT firstName
FROM People
WHERE pid IN (SELECT pid FROM Customers)
AND pid NOT IN (SELECT custId FROM Orders);
```

Grade: 9.5/10, better formatting could be done here

5. Show the first name of customers who have never placed an order. Use one inner and one outer join.

```
-- Me
SELECT p.firstName
FROM People p INNER JOIN Customers c on p.pid = c.pid
              LEFT OUTER JOIN Orders o on c.pid = o.custId
WHERE o.custId is NULL;
-- AI
SELECT p.firstName
FROM People p
```

```
JOIN Customers c ON p.pid = c.pid
LEFT JOIN Orders o ON c.pid = o.custId
WHERE o.custId IS NULL;
```

Grade: 9.5/10; Neatness counts!

6. Show the id and commission percent of Agents who booked an order for the Customer whose id is 007, sorted by commission percent from high to low. Use joins; no subqueries.

```
-- Me
SELECT DISTINCT a.pid, a.commissionPct as cPct
FROM Agents a INNER JOIN Orders o on a.pid = o.agentId
              INNER JOIN Customers c on o.custId = 007
ORDER BY cPct DESC;
-- AI
SELECT a.pid, a.commissionPct
FROM Agents a
JOIN Orders o ON a.pid = o.agentId
WHERE o.custId = 007
ORDER BY a.commissionPct DESC;
```

Grade: 9.5/10; I chose distinct as there was no reason to list an Agent more than once.

7. Show the last name, home city, and commission percent of Agents who booked an order for the customer whose id is 001, sorted by commission percent from high to low. Use joins.

```
-- Me
SELECT DISTINCT p.lastName, p.homeCity, a.commissionPct as cPct
FROM People p INNER JOIN Agents a on p.pid = a.pid
              INNER JOIN Orders o on a.pid = o.agentId
              INNER JOIN Customers c on o.custId = 001
ORDER BY cPct DESC;
-- AI
SELECT p.lastName, p.homeCity, a.commissionPct
FROM People p
JOIN Agents a ON p.pid = a.pid
JOIN Orders o ON a.pid = o.agentId
WHERE o.custId = 001
ORDER BY a.commissionPct DESC;
```

Grade: 9.5/10; I chose distinct as there was no reason to list an Agent more than once.

8. Show the last name and home city of customers who live in a city that makes the fewest different kinds of products. (Hint: Use count, group by, and having on the Products table in a subquery.)

```
-- Me
SELECT p.lastName, p.homeCity
```

```
FROM People p INNER JOIN Customers c on p.pid = c.pid
WHERE p.homeCity IN (
                     SELECT city
                     FROM Products
                     GROUP BY city
                     HAVING COUNT(prodId) = (
                                             SELECT MIN(num_prod)
                                             FROM (
                                                   SELECT COUNT(prodId) as
num_prod
                                                   FROM Products
                                                   GROUP BY city
                                                  )
                                            )
                    );
-- AI
SELECT p.lastName, p.homeCity
FROM People p
JOIN Customers c ON p.pid = c.pid
WHERE p.homeCity IN (
    SELECT city
    FROM Products
    GROUP BY city
    HAVING COUNT(DISTINCT prodId) = (
        SELECT MIN(product_count)
        FROM (
            SELECT COUNT(DISTINCT prodId) AS product_count
            FROM Products
            GROUP BY city
        ) AS counts
    )
);
```

Grade: 10/10. Nice formatting. I'm not sure what the 'AS counts' is for though. It also didn't need the DISTINCT due to PK constraints!

9. Show the name and id of all Products ordered through any Agent who booked at least one order for a Customer in Oyster Bay, sorted by product name from A to Z. You can use joins or subqueries. Better yet, impress me by doing it both ways.

```
-- Me: Subqueries
SELECT name, prodid
FROM Products
WHERE prodid IN (
                 SELECT prodid
                 FROM Orders
                 WHERE agentId IN (
                                   SELECT agentId
                                   FROM Orders
                                   WHERE custId IN (
                                                             SELECT pid
                                                             FROM People
                                                             WHERE homeCity =
'Oyster Bay'
                                                            )
                                      )
                    )
ORDER BY name ASC;
-- AI: Subqueries
SELECT name, prodId
FROM Products
WHERE prodId IN (
    SELECT prodId
    FROM Orders
    WHERE agentId IN (
        SELECT agentId
        FROM Orders
        WHERE custId IN (
            SELECT pid
            FROM People
            WHERE homeCity = 'Oyster Bay'
        )
    )
)
ORDER BY name;

-- Me: Joins
SELECT name, prodid
FROM Products
WHERE prodId IN (
                  SELECT o.prodId
                  FROM Orders o INNER JOIN Products p ON o.prodId =
p.prodId
                  WHERE o.agentId IN (
                                      SELECT a.pid
                                      FROM Agents a INNER JOIN Orders o
ON a.pid = o.agentId
                                      WHERE o.custId IN (
                                                          SELECT c.pid
                                                          FROM Customers
c INNER JOIN People p ON c.pid = p.pid
                                                          WHERE
p.homeCity = 'Oyster Bay'
                                                        )
                                     )
                )
ORDER BY name ASC;
-- AI: Joins attempt 1
SELECT DISTINCT pr.name, pr.prodId
FROM Products pr
JOIN Orders o ON pr.prodId = o.prodId
JOIN Agents a ON o.agentId = a.pid
JOIN Customers c ON o.custId = c.pid
JOIN People p ON c.pid = p.pid
```

```
WHERE p.homeCity = 'Oyster Bay'
ORDER BY pr.name;
-- AI: Joins attempt 2
SELECT DISTINCT pr.name, pr.prodId
FROM Products pr
JOIN Orders o ON pr.prodId = o.prodId
JOIN Agents a ON o.agentId = a.pid
WHERE a.pid IN (SELECT agentId FROM Orders WHERE custId IN (SELECT pid FROM
People WHERE homeCity = 'Oyster Bay'))
ORDER BY pr.name;
```

Grade: Subquery: 11/10. It has better formatting than I did 😮. Joins: 8/10. It missed the flux capacitor! Now that I inspect my response I notice that I still used subqueries. Whoops. It is intuitive to me do approach the problem in this manner as I can test different parts before clicking them together. Trying to debug the AI's response is difficult as I cannot pick each individual piece out of the equation. To add insult to injury, Gemini gaslit me into thinking I was wrong. When I provided it with the specifics of the missing row (flux capacitor), it tried again and got the right answer. It too resorted to subqueries.

10. Show the first and last name of customers and agents living in the same city, along with the name of their shared city. (Living in a city with yourself does not count, so exclude those from your results.)

```
-- Me
SELECT pc.firstName AS cust_first, pc.lastName AS cust_last, pa.firstName
AS agent_first, pa.lastName AS agent_last, pc.homeCity AS shared_city
FROM People pc INNER JOIN Customers c ON pc.pid = c.pid
               INNER JOIN People pa ON pa.homeCity = pc.homeCity
               INNER JOIN Agents a ON pa.pid = a.pid
WHERE pc.pid != pa.pid;
-- AI
SELECT pc.firstName AS customer_first, pc.lastName AS customer_last,
pa.firstName AS agent_first, pa.lastName AS agent_last, pc.homeCity
FROM People pc
JOIN Customers c ON pc.pid = c.pid
JOIN People pa ON pc.homeCity = pa.homeCity
JOIN Agents a ON pa.pid = a.pid
WHERE pc.pid <> pa.pid;
```

Grade: 12/10. I had to ask for help on this one. I was stuck in subquery mode (I plan to utilize them more than Joins due to the way that I think). It told me to alias the People twice! Smart. It used the '<>' operator instead of '!=' as it will be more portable. Nice touch I will admit.

Overall Grade: 10/10. AI is still powering through these!