SQL Exercise

For this SQL exercise, I began by reviewing the ERD to understand the relationships between the tables, including how Orders connect to Customers and how order-level details are stored. From there, I approached each question by identifying which tables contained the fields I needed, determining the correct join paths, and deciding what type of aggregation or filtering was required. I used PostgreSQL syntax for all queries and focused on writing clear, readable SQL with consistent formatting and aliases. Throughout the exercise, I made sure to explain my reasoning step-by-step so it is easy to follow how I arrived at each solution.

1. Approach:
For this question, I needed to count how many orders were placed in each month of 2019.

These were my steps:

1. I used the Orders table as it contains OrderDate.
2. I extracted the month from OrderDate and named it ordermonth.
3. I filtered to orders in 2019 by extracting the year from OrderDate.
4. I grouped by ordermonth to group the data into months.
5. I used COUNT() to count the number of orders in each group (ordercount).
6. I ordered the results from January to December using ORDER BY ordermonth.

Query:

```
SELECT
   EXTRACT(MONTH FROM OrderDate) AS ordermonth,
   COUNT(*) AS ordercount
FROM Orders
WHERE EXTRACT(YEAR FROM OrderDate) = 2019
GROUP BY ordermonth
ORDER BY ordermonth;
```

2. Approach:
For this question, I needed the customers located in California and the orders they placed.

These were my steps:

1. I used the **Customers** table as my primary table.
2. I selected the specific fields requested: CustomerID, first and last name, and OrderID.
3. I joined the Customers table to the Orders table on CustomerID, as it is the primary key in Customers and a foreign key in Orders.
4. I filtered to c.State = 'CA' so that only customers in California were returned.

Query:

```
SELECT
   c.CustomerID,
   c.FirstName,
   c.LastName,
   o.OrderID
FROM Customers AS c
JOIN Orders AS o
   ON c.CustomerID = o.CustomerID
WHERE c.State = 'CA';
```

3. Approach:

For this question, I had to calculate the total order amount, which is done by summing the cost of each line item (quantity × unit price).

These were my steps:

1. I used the Orders table to get OrderID and OrderDate.
2. I joined to OrderDetails to access UnitPrice and Quantity.
3. I calculated the total order amount as SUM(Quantity * UnitPrice) and aliased it as totalorderamount.
4. I filtered to orders in 2020 by extracting the year from OrderDate.
5. I grouped by OrderID because an order can contain multiple products in the OrderDetails table. Grouping ensures all line items for each order are summed together into one total.
6. I ordered the results from highest to lowest using ORDER BY totalorderamount DESC.

Query:

```
SELECT
   o.OrderID,
   SUM(od.Quantity * od.UnitPrice) AS totalorderamount
FROM Orders AS o
JOIN OrderDetails AS od
   ON o.OrderID = od.OrderID
WHERE EXTRACT(YEAR FROM o.OrderDate) = 2020
GROUP BY o.OrderID
ORDER BY totalorderamount DESC;
```

4. Approach**:**
For this question, I had to return the highest-priced product for each customer by the month of the order date. Since the highest price had to be evaluated within each customer and each month, I used a window function to rank products by their unit price.

These were my steps:

1. I joined:
    - Customers → Orders
    - Orders → OrderDetails

    - OrderDetails → Products
      so that I could access CustomerID, OrderDate, ProductName, and UnitPrice in one query.
2. I extracted the order month using EXTRACT(MONTH FROM o.OrderDate) because the question requires grouping by month.
3. I used a ROW_NUMBER() window function to rank products for each customer and month combination, ordering by UnitPrice in descending order so the highest-priced item gets rank 1.
4. I wrapped this logic in a CTE so I could easily filter for only the rows where rn = 1.
5. I selected the required fields and ordered the results by customer and month for readability.

Query:

```
WITH RankedItems AS (
  SELECT
    c.CustomerID,
    EXTRACT(MONTH FROM o.OrderDate) AS order_month,
    p.ProductName,
    od.UnitPrice,
    ROW_NUMBER() OVER (
      PARTITION BY c.CustomerID,
               EXTRACT(MONTH FROM o.OrderDate)
      ORDER BY od.UnitPrice DESC
    ) AS rn
  FROM Customers AS c
  JOIN Orders AS o
    ON c.CustomerID = o.CustomerID
  JOIN OrderDetails AS od
    ON o.OrderID = od.OrderID
  JOIN Products AS p
    ON od.ProductID = p.ProductID
)
SELECT
  CustomerID,
  order_month,
  ProductName,
  UnitPrice
FROM RankedItems
WHERE rn = 1
ORDER BY CustomerID, order_month;
```