SQL Query Collection

1. List all orders with customer names and product names

Explanation: Joins all three tables to show who ordered what and when.

```
SELECT
  o.OrderID,
  c.CustomerName,
  p.ProductName,
  o.OrderDate,
  o.Quantity
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID;
```

2. Find all customers who placed an order for 'Product_14'

Explanation: Filters orders to find customers who ordered a specific product.

```
SELECT DISTINCT c.CustomerName, c.City
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE p.ProductName = 'Product_14';
```

3. Show total quantity ordered per customer

Explanation: Aggregates quantity per customer.

```
SELECT
  c.CustomerName,
  SUM(o.Quantity) AS TotalQuantity
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.CustomerName;
```

4. List customers who have never placed an order

Explanation: Uses LEFT JOIN to find customers without matching orders.

```
SELECT
  c.CustomerName,
  c.City
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.CustomerID IS NULL;
```

5. Find products that have never been ordered

Explanation: Identifies products with no associated orders.

```
SELECT
  p.ProductName,
  p.Price
FROM Products p
LEFT JOIN Orders o ON p.ProductID = o.ProductID
WHERE o.ProductID IS NULL;
```

6. Show total revenue per product (Price × Quantity)

Explanation: Calculates revenue per product.

```
SELECT
  p.ProductName,
  SUM(p.Price * o.Quantity) AS TotalRevenue
FROM Orders o
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY p.ProductName;
```

7. List orders placed in the last 90 days with customer and product details

Explanation: Filters recent orders.

```
SELECT
  o.OrderID,
  c.CustomerName,
  p.ProductName,
  o.OrderDate,
  o.Quantity
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE o.OrderDate >= CURRENT_DATE - INTERVAL '90 DAY';
```

8. Find the most frequently ordered product

Explanation: Ranks products by order count.

```
SELECT
  p.ProductName,
  COUNT(o.OrderID) AS OrderCount
FROM Orders o
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY OrderCount DESC
LIMIT 1;
```

9. Show customers who ordered more than 3 different products

Explanation: Counts distinct products per customer.

```
SELECT
  c.CustomerName,
  COUNT(DISTINCT o.ProductID) AS ProductsOrdered
FROM Orders o
```

```
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.CustomerName
HAVING COUNT(DISTINCT o.ProductID) > 3;
```

10. List cities with total number of orders placed

Explanation: Aggregates orders by city.

```
SELECT
  c.City,
  COUNT(o.OrderID) AS TotalOrders
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.City;
```

11. Find customers who ordered products priced above 50

Explanation: Filters customers based on product price.

```
SELECT DISTINCT
  c.CustomerName,
  c.City
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE p.Price > 50;
```

12. Show products ordered by customers from 'Chicago'

Explanation: Filters products by customer city.

```
SELECT DISTINCT
  p.ProductName
FROM Orders o
JOIN Products p ON o.ProductID = p.ProductID
JOIN Customers c ON o.CustomerID = c.CustomerID
WHERE c.City = 'Chicago';
```

13. List customers who ordered the same product more than once

Explanation: Identifies repeat orders.

```
SELECT
  c.CustomerName,
  p.ProductName,
  COUNT(*) AS OrderCount
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY c.CustomerName, p.ProductName
HAVING COUNT(*) > 1;
```

14. Show the latest order date for each customer

Explanation: Finds most recent order per customer.

```
SELECT
  c.CustomerName,
  MAX(o.OrderDate) AS LatestOrder
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.CustomerName;
```

15. Find the top 3 customers by total spending

Explanation: Ranks customers by spending.

```
SELECT
  c.CustomerName,
  SUM(p.Price * o.Quantity) AS TotalSpent
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY c.CustomerName
ORDER BY TotalSpent DESC
LIMIT 3;
```

16. List products ordered by more than 5 different customers

Explanation: Counts unique customers per product.

```
SELECT
  p.ProductName,
  COUNT(DISTINCT o.CustomerID) AS CustomerCount
FROM Orders o
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY p.ProductName
HAVING COUNT(DISTINCT o.CustomerID) > 5;
```

17. Show average quantity ordered per product

Explanation: Calculates average quantity.

```
SELECT
  p.ProductName,
  AVG(o.Quantity) AS AvgQuantity
FROM Orders o
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY p.ProductName;
```

18. Find customers who ordered both 'Product_1' and 'Product_25'

Explanation: Filters customers who ordered both products.

```
SELECT c.CustomerName
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
```

```
WHERE p.ProductName IN ('Product_1', 'Product_25')
GROUP BY c.CustomerName
HAVING COUNT(DISTINCT p.ProductName) = 2;
```

19. List all orders with product price and total cost

Explanation: Calculates total cost per order.

```
SELECT
  o.OrderID,
  c.CustomerName,
  p.ProductName,
  p.Price,
  o.Quantity,
  (p.Price * o.Quantity) AS TotalCost
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID;
```

20. Show customers who ordered products from multiple cities (hypothetical)

Explanation: Assumes products have city info.

```
SELECT
  c.CustomerName
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY c.CustomerName
HAVING COUNT(DISTINCT p.City) > 1;
```

Subquery Practice Problems

1. Find customers who placed the highest quantity order

```
SELECT c.CustomerName
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
WHERE o.Quantity = (SELECT MAX(Quantity) FROM Orders);
```

2. List products with price above the average product price

```
SELECT ProductName, Price
FROM Products
WHERE Price > (SELECT AVG(Price) FROM Products);
```

3. Show customers whose total spending is above the average spending

```
SELECT c.CustomerName, SUM(p.Price * o.Quantity) AS TotalSpent
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY c.CustomerName
```

```
HAVING SUM(p.Price * o.Quantity) > (
    SELECT AVG(Total)
    FROM (
        SELECT SUM(p2.Price * o2.Quantity) AS Total
        FROM Orders o2
        JOIN Products p2 ON o2.ProductID = p2.ProductID
        GROUP BY o2.CustomerID
    ) AS AvgSpending
);
```

4. Find the most expensive product ordered

```
SELECT ProductName, Price
FROM Products
WHERE Price = (SELECT MAX(Price) FROM Products);
```

5. List customers who ordered the cheapest product

```
SELECT DISTINCT c.CustomerName
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE p.Price = (SELECT MIN(Price) FROM Products);
```