

-- SQL Practice Dataset Solutions

-- Based on Customers, Products, and Orders tables provided.

-- --- Basic SELECT and Filtering ---

-- 1. Retrieve all customers.

```
SELECT * FROM Customers;
```

-- 2. List all products with price above \$50.

```
SELECT * FROM Products WHERE price > 50.00;
```

-- 3. Show all orders made in June 2025.

```
SELECT * FROM Orders WHERE order_date BETWEEN '2025-06-01' AND '2025-06-30';
```

-- 4. Retrieve customer names and countries.

```
SELECT name, country FROM Customers;
```

-- 5. List unique product categories.

```
SELECT DISTINCT category FROM Products;
```

-- 9. List customers who are older than 30.

```
SELECT * FROM Customers WHERE age > 30;
```

-- 10. Show orders with quantity greater than 2.

```
SELECT * FROM Orders WHERE quantity > 2;
```

-- 15. List products in 'Fitness' category.

```
SELECT * FROM Products WHERE category = 'Fitness';
```

-- 16. Show customers from India.

```
SELECT * FROM Customers WHERE country = 'India';
```

-- 29. Retrieve products with price between \$20 and \$100.

```
SELECT * FROM Products WHERE price BETWEEN 20.00 AND 100.00;
```

-- 38. Retrieve orders where customers ordered more than 3 units.

```
SELECT * FROM Orders WHERE quantity > 3;
```

-- ---- Aggregation and Grouping ---

-- 6. Find the number of customers from each country.

```
SELECT country, COUNT(customer_id) AS num_customers  
FROM Customers  
GROUP BY country;
```

-- 7. Calculate the average price of products.

```
SELECT AVG(price) AS average_price  
FROM Products;
```

-- 8. Show the highest priced product.

```
SELECT name, price  
FROM Products  
ORDER BY price DESC  
LIMIT 1;
```

-- 11. Count total number of orders.

```
SELECT COUNT(order_id) AS total_orders  
FROM Orders;
```

-- 12. Find the total quantity ordered for each product.

```
SELECT product_id, SUM(quantity) AS total_quantity_ordered
```

```
FROM Orders
```

```
GROUP BY product_id;
```

```
-- 17. Find the youngest customer.
```

```
SELECT *
```

```
FROM Customers
```

```
ORDER BY age ASC
```

```
LIMIT 1;
```

```
-- 22. Find the average age of customers by country.
```

```
SELECT country, AVG(age) AS average_age
```

```
FROM Customers
```

```
GROUP BY country;
```

```
-- 25. Find the total number of distinct products ordered.
```

```
SELECT COUNT(DISTINCT product_id) AS distinct_products_ordered
```

```
FROM Orders;
```

```
-- 28. Find the top 3 most expensive products.
```

```
SELECT name, price
```

```
FROM Products
```

```
ORDER BY price DESC
```

```
LIMIT 3;
```

```
-- 30. List the number of orders each customer has made.
```

```
SELECT customer_id, COUNT(order_id) AS total_orders
```

```
FROM Orders
```

```
GROUP BY customer_id;
```

```
-- 34. Find the total revenue generated in June 2025.
```

```
SELECT SUM(O.quantity * P.price) AS total_revenue
```

```
FROM Orders O
JOIN Products P ON O.product_id = P.product_id
WHERE O.order_date BETWEEN '2025-06-01' AND '2025-06-30';
```

-- 37. Find the most frequently ordered product (by total quantity).

```
SELECT P.name, SUM(O.quantity) AS total_quantity_sold
FROM Orders O
JOIN Products P ON O.product_id = P.product_id
GROUP BY P.name
ORDER BY total_quantity_sold DESC
LIMIT 1;
```

-- 39. Show the customer who has placed the highest number of orders.

```
SELECT C.name, COUNT(O.order_id) AS num_orders
FROM Customers C
JOIN Orders O ON C.customer_id = O.customer_id
GROUP BY C.name
ORDER BY num_orders DESC
LIMIT 1;
```

-- 42. List the highest priced product in each category. (Using Window Function)

```
SELECT category, name, price
FROM (
    SELECT
        *,
        RANK() OVER (PARTITION BY category ORDER BY price DESC) as rn
    FROM Products
) AS ranked_products
WHERE rn = 1;
```

-- 43. Retrieve the last order date for each customer.

```
SELECT customer_id, MAX(order_date) AS last_order_date
FROM Orders
GROUP BY customer_id;
```

-- 45. Show a list of products and the total quantity sold.

```
SELECT P.name, COALESCE(SUM(O.quantity), 0) AS total_quantity_sold
FROM Products P
LEFT JOIN Orders O ON P.product_id = O.product_id
GROUP BY P.name
ORDER BY total_quantity_sold DESC;
```

-- 48. Show products with the lowest total sales quantity.

```
SELECT P.name, COALESCE(SUM(O.quantity), 0) AS total_quantity_sold
FROM Products P
LEFT JOIN Orders O ON P.product_id = O.product_id
GROUP BY P.name
ORDER BY total_quantity_sold ASC;
```

-- 49. Retrieve the first order made by each customer. (Using Window Function)

```
SELECT O.order_id, C.name AS customer_name, O.order_date
FROM (
    SELECT
        *,
        RANK() OVER (PARTITION BY customer_id ORDER BY order_date ASC) as rn
    FROM Orders
) AS ranked_orders O
JOIN Customers C ON O.customer_id = C.customer_id
WHERE rn = 1;
```

-- --- Joining and Subqueries ---

-- 13. Find the total revenue for each product (quantity * price).

```
SELECT P.name, SUM(O.quantity * P.price) AS product_revenue  
FROM Orders O  
JOIN Products P ON O.product_id = P.product_id  
GROUP BY P.name;
```

-- 14. Retrieve customers who ordered 'Yoga Mat'.

```
SELECT DISTINCT C.name  
FROM Customers C  
JOIN Orders O ON C.customer_id = O.customer_id  
JOIN Products P ON O.product_id = P.product_id  
WHERE P.name = 'Yoga Mat';
```

-- 18. List products not ordered by any customer.

```
SELECT P.name  
FROM Products P  
LEFT JOIN Orders O ON P.product_id = O.product_id  
WHERE O.order_id IS NULL;
```

-- 19. Retrieve customers who ordered more than one distinct product.

```
SELECT C.name  
FROM Customers C  
JOIN Orders O ON C.customer_id = O.customer_id  
GROUP BY C.name  
HAVING COUNT(DISTINCT O.product_id) > 1;
```

-- 20. Find the customer who has spent the most.

```
SELECT C.name, SUM(O.quantity * P.price) AS total_spent  
FROM Customers C  
JOIN Orders O ON C.customer_id = O.customer_id
```

```
JOIN Products P ON O.product_id = P.product_id  
GROUP BY C.name  
ORDER BY total_spent DESC  
LIMIT 1;
```

-- 21. List customers who have not placed any orders.

```
SELECT C.name  
FROM Customers C  
LEFT JOIN Orders O ON C.customer_id = O.customer_id  
WHERE O.order_id IS NULL;
```

-- 23. Retrieve products ordered by customers from the USA.

```
SELECT DISTINCT P.name  
FROM Products P  
JOIN Orders O ON P.product_id = O.product_id  
JOIN Customers C ON O.customer_id = C.customer_id  
WHERE C.country = 'USA';
```

-- 24. List customers who ordered products in 'Home Kitchen' category.

```
SELECT DISTINCT C.name  
FROM Customers C  
JOIN Orders O ON C.customer_id = O.customer_id  
JOIN Products P ON O.product_id = P.product_id  
WHERE P.category = 'Home Kitchen';
```

-- 27. Show customers and their total spent amount.

```
SELECT C.name, COALESCE(SUM(O.quantity * P.price), 0) AS total_spent  
FROM Customers C  
LEFT JOIN Orders O ON C.customer_id = O.customer_id  
LEFT JOIN Products P ON O.product_id = P.product_id  
GROUP BY C.name;
```

-- 31. Show orders including customer names and product names.

```
SELECT  
    O.order_id,  
    C.name AS customer_name,  
    P.name AS product_name,  
    O.quantity,  
    O.order_date  
FROM Orders O  
JOIN Customers C ON O.customer_id = C.customer_id  
JOIN Products P ON O.product_id = P.product_id;
```

-- 36. List orders along with the product category.

```
SELECT  
    O.order_id,  
    P.name AS product_name,  
    P.category,  
    O.quantity,  
    O.order_date  
FROM Orders O  
JOIN Products P ON O.product_id = P.product_id;
```

-- 40. List customers with the highest average order quantity.

```
SELECT C.name, AVG(O.quantity) AS average_quantity  
FROM Customers C  
JOIN Orders O ON C.customer_id = O.customer_id  
GROUP BY C.name  
ORDER BY average_quantity DESC  
LIMIT 1;
```

-- 41. Find customers who have never ordered products over \$50.

```
SELECT name
FROM Customers
WHERE customer_id NOT IN (
    SELECT DISTINCT O.customer_id
    FROM Orders O
    JOIN Products P ON O.product_id = P.product_id
    WHERE P.price > 50.00
);
```

-- 44. Find customers who ordered 'Running Shoes'.

```
SELECT DISTINCT C.name
FROM Customers C
JOIN Orders O ON C.customer_id = O.customer_id
JOIN Products P ON O.product_id = P.product_id
WHERE P.name = 'Running Shoes';
```

-- 46. Retrieve orders along with the customer's age.

```
SELECT O.*, C.age
FROM Orders O
JOIN Customers C ON O.customer_id = C.customer_id;
```

-- 50. List all customers and indicate if they have placed an order or not.

```
SELECT
    C.name,
    CASE WHEN O.customer_id IS NOT NULL THEN 'Yes' ELSE 'No' END AS placed_order
FROM Customers C
LEFT JOIN Orders O ON C.customer_id = O.customer_id
GROUP BY C.name;
```

-- --- Advanced Queries ---

-- 26. List customers who have ordered on consecutive days.

```
SELECT DISTINCT C.name  
FROM Customers C  
JOIN Orders O1 ON C.customer_id = O1.customer_id  
JOIN Orders O2 ON C.customer_id = O2.customer_id  
WHERE O2.order_date = DATE_ADD(O1.order_date, INTERVAL 1 DAY);
```

-- 32. Find customers who have only ordered 'Electronics' products.

```
SELECT C.name  
FROM Customers C  
JOIN Orders O ON C.customer_id = O.customer_id  
JOIN Products P ON O.product_id = P.product_id  
GROUP BY C.name  
HAVING SUM(CASE WHEN P.category != 'Electronics' THEN 1 ELSE 0 END) = 0;
```

-- 33. List products ordered by customers younger than 30.

```
SELECT DISTINCT P.name  
FROM Products P  
JOIN Orders O ON P.product_id = O.product_id  
JOIN Customers C ON O.customer_id = C.customer_id  
WHERE C.age < 30;
```

-- 35. Retrieve customers who have ordered from at least two categories.

```
SELECT C.name  
FROM Customers C  
JOIN Orders O ON C.customer_id = O.customer_id  
JOIN Products P ON O.product_id = P.product_id  
GROUP BY C.name  
HAVING COUNT(DISTINCT P.category) >= 2;
```

-- 47. Find customers who have ordered products from three different categories.

```
SELECT C.name  
FROM Customers C  
JOIN Orders O ON C.customer_id = O.customer_id  
JOIN Products P ON O.product_id = P.product_id  
GROUP BY C.name  
HAVING COUNT(DISTINCT P.category) = 3;
```