```
-- SQL Select
-- Question: Show the first 10 customers sorted alphabetically by name
SELECT customer_name, email
FROM customers
ORDER BY customer_name
LIMIT 10;

-- SQL Select Distinct
-- Question: List unique cities where customers are located
SELECT DISTINCT city
FROM customers;

-- SQL Where
-- Question: Find customers who signed up after January 1, 2025
SELECT customer_name, signup_date
FROM customers
WHERE signup_date > '2025-01-01';

-- SQL Order By
-- Question: List products by stock quantity in ascending order, then by price descending
SELECT product_name, stock_quantity, price
```

```
FROM products
ORDER BY stock_quantity ASC, price DESC;

-- SQL And
-- Question: Find orders placed in April 2025 with total amount over 500
SELECT order_id, order_date, total_amount
FROM orders
WHERE order_date LIKE '2025-04%' AND total_amount > 500;

-- SQL Or
-- Question: Show products that are either out of stock or priced below 10
SELECT product_name, price, stock_quantity
FROM products
WHERE stock_quantity = 0 OR price < 10;

-- SQL Not
-- Question: Find customers whose email does not end with '.com'
SELECT customer_name, email
FROM customers
WHERE email NOT LIKE '%.com';

-- SQL Insert Into
```

```
-- Question: Add multiple new products to the Clothing category
INSERT INTO products (product_name, category_id, price,
stock_quantity)
VALUES
   ('Jacket', 2, 89.99, 60),
   ('Sneakers', 2, 59.99, 40);

-- SQL Null Values
-- Question: Count customers with missing postal codes
SELECT COUNT(*) AS missing_postal_codes
FROM customers
WHERE postal_code IS NULL;

-- SQL Update
-- Question: Set stock quantity to 0 for products with price below 15
UPDATE products
SET stock_quantity = 0
WHERE price < 15;

-- SQL Delete
-- Question: Delete order details for orders with zero quantity
DELETE FROM order_details
WHERE quantity = 0;
```

```
-- SQL Select Top
-- Question: Get the top 5 customers by total order amount
SELECT c.customer_name, SUM(o.total_amount) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.customer_name
ORDER BY total_spent DESC
LIMIT 5;

-- SQL Aggregate Functions: Min and Max
-- Question: Find the earliest and latest order dates
SELECT MIN(order_date) AS earliest_order, MAX(order_date) AS
latest_order
FROM orders;

-- SQL Count
-- Question: Count the number of orders per category
SELECT c.category_name, COUNT(od.order_id) AS order_count
FROM categories c
JOIN products p ON c.category_id = p.category_id
JOIN order_details od ON p.product_id = od.product_id
GROUP BY c.category_name;

-- SQL Sum
```

```
-- Question: Calculate the total quantity of products ordered
SELECT SUM(quantity) AS total_items_ordered
FROM order_details;

-- SQL Avg
-- Question: Find the average order amount per customer
SELECT c.customer_name, AVG(o.total_amount) AS avg_order_value
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_name;

-- SQL Like
-- Question: Find products with names containing 'top'
SELECT product_name
FROM products
WHERE product_name LIKE '%top%';

-- SQL Wildcards
-- Question: Find customers with email addresses ending in '.org' or '.edu'
SELECT customer_name, email
FROM customers
WHERE email LIKE '%.org' OR email LIKE '%.edu';
```

```
-- SQL In
-- Question: List products in Electronics or Clothing categories
SELECT product_name, c.category_name
FROM products p
JOIN categories c ON p.category_id = c.category_id
WHERE c.category_name IN ('Electronics', 'Clothing');

-- SQL Between
-- Question: Find orders placed between April 1 and April 15, 2025
SELECT order_id, order_date
FROM orders
WHERE order_date BETWEEN '2025-04-01' AND '2025-04-15';

-- SQL Aliases
-- Question: Show order details with product and customer aliases
SELECT o.order_id AS OrderNumber, c.customer_name AS Buyer,
p.product_name AS Item
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
JOIN products p ON od.product_id = p.product_id
JOIN customers c ON o.customer_id = c.customer_id;

-- SQL Joins: Inner Join
-- Question: Get product names and quantities for a specific order
```

```
SELECT p.product_name, od.quantity
FROM products p
INNER JOIN order_details od ON p.product_id = od.product_id
WHERE od.order_id = 1;

-- SQL Left Join
-- Question: List all products and their order quantities, including products not ordered
SELECT p.product_name, SUM(od.quantity) AS total_ordered
FROM products p
LEFT JOIN order_details od ON p.product_id = od.product_id
GROUP BY p.product_name;

-- SQL Right Join
-- Question: Show all orders and their shippers, including unused shippers
SELECT o.order_id, s.shipper_name
FROM orders o
RIGHT JOIN shippers s ON o.shipper_id = s.shipper_id;

-- SQL Full Join (Emulated with UNION)
-- Question: Combine all products and order details, showing all matches and non-matches
SELECT p.product_name, od.quantity
FROM products p
```

```
LEFT JOIN order_details od ON p.product_id = od.product_id
UNION
SELECT p.product_name, od.quantity
FROM products p
RIGHT JOIN order_details od ON p.product_id = od.product_id;

-- SQL Self Join
-- Question: Find products with similar prices (within $10)
SELECT p1.product_name, p2.product_name, p1.price, p2.price
FROM products p1
JOIN products p2 ON p1.product_id < p2.product_id
WHERE ABS(p1.price - p2.price) <= 10;

-- SQL Union
-- Question: Combine customer emails and shipper phone numbers
SELECT email AS contact_info
FROM customers
UNION
SELECT phone
FROM shippers;

-- SQL Group By
-- Question: Group orders by year and count them
```

```
SELECT YEAR(order_date) AS order_year, COUNT(order_id) AS
order_count
FROM orders
GROUP BY YEAR(order_date);

-- SQL Having
-- Question: Find customers who spent more than 1000 in total
SELECT c.customer_name, SUM(o.total_amount) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_name
HAVING SUM(o.total_amount) > 1000;

-- SQL Exists
-- Question: Find products that have been ordered
SELECT product_name
FROM products p
WHERE EXISTS (
   SELECT 1
   FROM order_details od
   WHERE od.product_id = p.product_id
);

-- SQL Any, All
-- Question: Find products cheaper than all products in Electronics
```

```
SELECT product_name, price
FROM products
WHERE price < ALL (
   SELECT price
   FROM products p
   JOIN categories c ON p.category_id = c.category_id
   WHERE c.category_name = 'Electronics'
);

-- SQL Select Into (Emulated with CREATE TABLE AS)
-- Question: Create a table with products under $50
CREATE TABLE budget_products AS
SELECT product_name, price
FROM products
WHERE price < 50;

-- SQL Insert Into Select
-- Question: Copy Electronics products into a new table
CREATE TABLE electronics_products AS
SELECT p.product_name, p.price
FROM products p
JOIN categories c ON p.category_id = c.category_id
WHERE c.category_name = 'Electronics';

-- SQL Case
-- Question: Classify orders by size based on total amount
```

```
SELECT order_id, total_amount,
   CASE
      WHEN total_amount < 100 THEN 'Small'
      WHEN total_amount <= 1000 THEN 'Medium'
      ELSE 'Large'
   END AS order_size
FROM orders;

-- SQL Null Functions
-- Question: Replace NULL stock quantities with 0
SELECT product_name, IFNULL(stock_quantity, 0) AS available_stock
FROM products;

-- SQL Stored Procedures
-- Question: Create a procedure to update product stock
DELIMITER //
CREATE PROCEDURE UpdateStock(IN prod_id INT, IN new_stock INT)
BEGIN
   UPDATE products
   SET stock_quantity = new_stock
   WHERE product_id = prod_id;
END //
DELIMITER ;

-- SQL Comments
-- Question: Add detailed comments to a query
```

```
/*
 * Query to find top-selling products
 * Joins products with order details to calculate total quantity sold
 */
SELECT p.product_name, SUM(od.quantity) AS total_sold -- Total units sold
FROM products p
JOIN order_details od ON p.product_id = od.product_id
GROUP BY p.product_name
ORDER BY total_sold DESC;

-- SQL Operators
-- Question: Find products with price 20% above average
SELECT product_name, price
FROM products
WHERE price > (SELECT AVG(price) * 1.2 FROM products);

-- SQL Create Table
-- Question: Create a table for promotions
CREATE TABLE promotions (
    promotion_id INT PRIMARY KEY AUTO_INCREMENT,
    product_id INT,
    discount_percentage DECIMAL(5,2),
    start_date DATE,
    end_date DATE,
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

```
-- SQL Drop Table
-- Question: Remove the promotions table
DROP TABLE IF EXISTS promotions;

-- SQL Alter Table
-- Question: Add a column for product ratings
ALTER TABLE products
ADD average_rating DECIMAL(3,1) DEFAULT 0;

-- SQL Constraints
-- Question: Create a table with multiple constraints
CREATE TABLE discounts (
    discount_id INT PRIMARY KEY AUTO_INCREMENT,
    discount_name VARCHAR(50) NOT NULL,
    percentage DECIMAL(5,2) CHECK (percentage BETWEEN 0 AND 100),
    product_id INT UNIQUE,
    active BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);

-- SQL Index
-- Question: Create a composite index on order_date and total_amount
CREATE INDEX idx_order_date_amount
ON orders(order_date, total_amount);
```

```
-- SQL Auto Increment
-- Question: Create a table with auto-incrementing IDs for logs
CREATE TABLE audit_logs (
    log_id INT PRIMARY KEY AUTO_INCREMENT,
    action VARCHAR(100),
    action_date DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- SQL Dates
-- Question: Find orders from the last 7 days
SELECT order_id, order_date
FROM orders
WHERE order_date >= DATE_SUB(CURRENT_DATE, INTERVAL 7 DAY);

-- SQL Views
-- Question: Create a view for customer order summaries
CREATE VIEW customer_order_summary AS
SELECT c.customer_name, COUNT(o.order_id) AS order_count,
SUM(o.total_amount) AS total_spent
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_name;

-- SQL Injection (Example of safe query)
-- Question: Safely retrieve customer by email (parameterized query example)
-- Note: Actual implementation depends on the application language
SELECT customer_name, email
```

```
FROM customers
WHERE email = ?; -- Placeholder for parameterized input

-- SQL Data Types
-- Question: Create a table with diverse data types
CREATE TABLE user_profiles (
    profile_id INT PRIMARY KEY,
    username VARCHAR(50),
    balance DECIMAL(10,2),
    last_login TIMESTAMP,
    is_verified BOOLEAN,
    bio TEXT
);

-- MySQL Functions
-- Question: Format order dates as 'Month Day, Year'
SELECT order_id, DATE_FORMAT(order_date, '%M %d, %Y') AS formatted_date
FROM orders;

-- SQL Server Functions (MySQL equivalent)
-- Question: Get the length of product names
SELECT product_name, LENGTH(product_name) AS name_length
FROM products;

-- MS Access Functions (MySQL equivalent)
-- Question: Round product prices to 1 decimal place
```

```
SELECT product_name, ROUND(price, 1) AS rounded_price
FROM products;
```