Assumptions Dataset

customers (customer_id, name, email, country)

orders (order_id, customer_id, order_date, status)

order_items (order_item_id, order_id, product_id, quantity, price)

products (product_id, name, category, stock)

1. Basic SELECT + WHERE + ORDER BY

```
-- Get all orders from USA sorted by order_date
SELECT o.order_id, o.order_date, c.name, c.country
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
WHERE c.country = 'USA'
ORDER BY o.order_date DESC;
```

2. GROUP BY + Aggregate Function

```
-- Total orders by country
SELECT c.country, COUNT(o.order_id) AS total_orders
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.country
ORDER BY total_orders DESC;
```

3. JOINs (INNER, LEFT, RIGHT)sql CopyEdit

```
-- INNER JOIN: Get all orders with product details
SELECT o.order_id, p.name AS product_name, oi.quantity, oi.price
FROM orders o
INNER JOIN order_items oi ON o.order_id = oi.order_id
INNER JOIN products p ON oi.product_id = p.product_id;

-- LEFT JOIN: Get all products and their sales (if any)
SELECT p.product_id, p.name, SUM(oi.quantity) AS total_sold
FROM products p
LEFT JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_id;

-- RIGHT JOIN: Get all orders and products (some products may not be ordered)
SELECT o.order_id, p.name AS product_name
FROM order_items oi
```

```
    RIGHT JOIN products p ON oi.product_id = p.product_id
    RIGHT JOIN orders o ON oi.order_id = o.order_id;
```

4. Subquery

```
 -- Products with sales above average
SELECT name, product_id
FROM products
WHERE product_id IN (
  SELECT product_id
  FROM order_items
  GROUP BY product_id
  HAVING SUM(quantity) > (
    SELECT AVG(total_quantity)
    FROM (
      SELECT SUM(quantity) AS total_quantity
      FROM order_items
      GROUP BY product_id
    ) sub
  )
);
```

5. Aggregate Functions (SUM, AVG)

```
 -- Total revenue per product
SELECT p.name, SUM(oi.quantity * oi.price) AS revenue
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.name
ORDER BY revenue DESC;
```

6.Create a View

```
-- View to show total revenue per customer

CREATE VIEW customer_revenue AS

SELECT c.customer_id, c.name, SUM(oi.quantity * oi.price) AS total_spent

FROM customers c

JOIN orders o ON c.customer_id = o.customer_id

JOIN order_items oi ON o.order_id = oi.order_id

GROUP BY c.customer_id;
```

7.Optimize with Index

```
-- Create indexes to speed up JOINs and WHERE filters

CREATE INDEX idx_orders_customer_id ON orders(customer_id);

CREATE INDEX idx_order_items_order_id ON order_items(order_id);

CREATE INDEX idx_order_items_product_id ON order_items(product_id);
```