SQL CHALLENGES FOR AN ECOMMERCE DATA:

Challenge 1: Inventory Management

Write queries to:

1. 1. Identify products that need reordering across all warehouses

```
SELECT
  p.product_id,
  p.product_name,
  SUM(i.quantity) AS total_stock,
  p.reorder_level
FROM products p
LEFT JOIN inventory i ON i.product_id = p.product_id
GROUP BY p.product_id, p.product_name, p.reorder_level
HAVING SUM(i.quantity) < p.reorder_level;
```

2. 2. Calculate the cost of restocking all products below reorder level

```
SELECT
  p.product_id,
  p.product_name,
  p.cost_price,
  p.reorder_level,
  SUM(i.quantity) AS total_stock,
  (p.reorder_level - SUM(i.quantity)) * p.cost_price AS restock_cost
FROM products p
LEFT JOIN inventory i ON i.product_id = p.product_id
GROUP BY p.product_id, p.product_name, p.cost_price, p.reorder_level
HAVING SUM(i.quantity) < p.reorder_level;
```

3. 3. Recommend warehouse transfers to balance inventory

```
SELECT
  product_id,
  warehouse_id,
  quantity
FROM inventory
ORDER BY product_id, quantity DESC;
```

Challenge 2: Customer Analytics

4. 1. Create a customer cohort analysis by registration month

```
SELECT
  DATE_TRUNC('month', registration_date) AS cohort_month,
  COUNT(*) AS total_customers
FROM customers
GROUP BY DATE_TRUNC('month', registration_date)
```

```
ORDER BY cohort_month;
```

5. 2. Calculate customer churn rate

```
SELECT
  c.customer_id,
  c.first_name,
  c.last_name
FROM customers c
LEFT JOIN orders o
  ON o.customer_id = c.customer_id
  AND o.order_date >= NOW() - INTERVAL '90 days'
WHERE o.order_id IS NULL;
```

6. 3. Identify customers likely to upgrade loyalty tiers

```
SELECT
  customer_id,
  first_name,
  last_name,
  loyalty_tier,
  total_spent
FROM customers
ORDER BY total_spent DESC;
```

Challenge 3: Revenue Optimization

7. 1. Find the most profitable product combinations

```
SELECT
  p.product_id,
  p.product_name,
  SUM(oi.subtotal) AS revenue
FROM order_items oi
JOIN products p ON p.product_id = oi.product_id
GROUP BY p.product_id, p.product_name
ORDER BY revenue DESC;
```

8. 2. Analyze discount effectiveness on revenue

```
SELECT
  discount,
  COUNT(*) AS times_used,
  SUM(subtotal) AS total_revenue
FROM order_items
GROUP BY discount
ORDER BY discount;
```

9. 3. Calculate revenue per warehouse

```
SELECT
  w.warehouse_id,
  w.warehouse_name,
  SUM(oi.subtotal) AS revenue
FROM shipments s
JOIN warehouses w ON w.warehouse_id = s.warehouse_id
JOIN orders o ON o.order_id = s.order_id
JOIN order_items oi ON oi.order_id = o.order_id
GROUP BY w.warehouse_id, w.warehouse_name
ORDER BY revenue DESC;
```

Challenge 4: Performance Tuning

10. 1. Optimize the slowest queries using EXPLAIN

```
EXPLAIN ANALYZE
SELECT * FROM products;
```

11. 2. Create appropriate indexes

```
CREATE INDEX idx_products_category ON products (category_id);
CREATE INDEX idx_orders_customer ON orders (customer_id);
CREATE INDEX idx_inventory_product ON inventory (product_id);
```

12. 3. Rewrite a subquery using JOINs for better performance

Before the subquery:

```
SELECT product_name
FROM products
WHERE product_id IN (SELECT product_id FROM inventory);
```

After join:

```
SELECT DISTINCT p.product_name
FROM products p
JOIN inventory i ON i.product_id = p.product_id;
```