**Joins & Unions**

1. List every order, showing its ID, date, customer name, store city and country, plus for each order line the product name and quantity—include orders even if they have no items yet.
2. Find all pairs of products that share the same brand but belong to different categories; show both product names, their categories, and the brand name.
3. Produce a single list of all distinct “locations” where the business operates, combining store cities and customer cities, with a column labeling each row as ‘Store’ or ‘Customer’.

**Subqueries**

1. Retrieve all products whose list price exceeds the average list price of their category.
2. For each store, show its name and the number of orders it processed that exceeded that store’s own average order total.
3. List customers who have placed more orders than the average number of orders per customer.

**CASE**

1. For each order, calculate its total amount and then categorize it as ‘Low’, ‘Medium’ or ‘High’ value using thresholds of your choice.
2. List every product and assign a stock-status label—‘Out of Stock’, ‘Low Stock’ (<10 units), or ‘In Stock’—based on its total quantity across all stores.

**IF…ELSE (T‑SQL)**

1. Write a T‑SQL script that takes a @StoreID parameter: if that store has processed zero orders in the last 30 days, print “No recent sales”; otherwise print “Active store”.
2. Build an IF…ELSE block that accepts @BrandID and prints the number of products under that brand, or “Brand not found” if none exist.

**Views**

1. Create a view showing each customer’s total lifetime spend and most recent order date.
2. Define a view that returns each product’s total sold quantity and average discount received.

**Indexes**

1. Identify a query that filters orders by order date and customer city. Then describe what non‑clustered index you’d create to optimize it (no need to write the DDL).
2. Propose an index strategy to speed up retrieving the top‑selling products in a given month.

**Triggers**

1. Create an AFTER INSERT trigger on order\_items that updates a running total in a separate product\_sales\_summary table.
2. Write a trigger on stocks that prevents any update setting quantity to a negative number—if attempted, roll back the operation.

**Stored Procedures**

1. Design a stored procedure usp\_GetCustomerOrders that takes @CustomerID and returns all their orders with totals and status.
2. Build a proc usp\_AdjustStock that accepts @StoreID, @ProductID, and @AdjustmentQty; it should apply the change and log it into an inventory\_log table.

**User‑Defined Functions**

1. Write a scalar UDF ufn\_CalcOrderTotal(@OrderID) returning the total amount (after discount) for that order.
2. Create a table-valued UDF ufn\_ProductsByCategory(@CategoryID) listing all products in a category along with their current total stock across all stores.

Happy practicing—and good luck on your test!

Here are 20 challenging questions based on the bikestores schema and your specified topics:

1. List customers who have placed orders in all stores located in their city.

2. Identify products that have never been stocked in any store but exist in the product catalog.

3. Find the top 3 staff members (by total orders handled) who managed orders requiring shipping within the same state as their store.

4. Calculate the total revenue difference between the current order and the previous order for each customer.

5. Create a view showing product names, their brands, categories, and average stock quantity across all stores.

6. Find orders where the discount applied is higher than the average discount for that product category.

7. Rank stores based on the number of unique products sold, even if some products have never been sold.

8. Write a trigger to prevent deleting a product if it exists in any unshipped order.

9. Retrieve customers who have ordered the same product in consecutive months.

10. Identify brands where all their products are priced above the overall median product price.

11. Design a stored procedure to transfer stock of a product from one store to another and update the inventory.

12. Find stores that have sold at least one product from every brand available in their stock.

13. Generate a report showing monthly sales growth/decline percentages for each store.

14. Create a UDF to return the total discounted revenue for a given customer between two dates.

15. List staff members who are managers and have not processed any orders in the last six months.

16. For each product category, find the month with the highest sales and compare it to the category’s annual average.

17. Use a union to combine active staff from all stores with inactive staff who have processed more than 50 orders.

18. Identify products that have been restocked (stock increased) after being completely sold out in a store.

19. Assign a dense rank to customers based on their total spending, partitioned by their state.

20. Find orders where the total item quantity exceeds the average quantity of all orders placed in the same store.

These questions require combining multiple concepts and careful analysis of the schema. Good luck with your test preparation!