**Inventory Systems**

**🔹 1. SELECT Statements (Basic to Mid) (5)**

1. Retrieve the list of products with their category names and prices above ₹500.
2. Show all purchases made by each supplier, including the product name and purchase date.
3. List all customers who have given a product review along with the product name and their rating.
4. Get the full contact information of users who handled sales.
5. Show the top 10 most recent inventory log entries including the product name, warehouse name, and action.

**🔹 2. Aggregate Functions (5)**

1. Find the total sales amount generated by each user.
2. Calculate the average price of products in each category.
3. Find the total quantity of each product purchased from suppliers.
4. Get the maximum rating given in product reviews for each product.
5. Count how many actions of type 'IN' and 'OUT' were done in the Inventory\_Log.

**🔹 3. GROUP BY + HAVING (5)**

1. Show the suppliers who supplied more than 5 distinct products.
2. Get each product's total quantity purchased where the total quantity is more than 100 units.
3. Show categories with an average product price above ₹1000.
4. List users who performed more than 10 purchases.
5. Find customers who wrote more than 2 reviews and gave an average rating below 3.

**🔹 4. JOINS (INNER, LEFT, RIGHT, FULL) (5)**

1. List all products along with their current stock from all warehouses (even if stock is zero).
2. Show sales details along with customer name and product name.
3. Get all suppliers and the total cost of purchases from them (even if no purchases).
4. Find users and their roles, showing all users even if they don’t have roles assigned.
5. List all products and include purchase details if available (use full outer join logic).

**🔹 5. Subqueries (Scalar, Correlated, Nested, IN, EXISTS) (6)**

1. List all products whose price is above the average product price.
2. Show all customers who haven’t made any sales.
3. Get the name of the product(s) with the highest stock quantity across all warehouses.
4. List all purchases where the price is greater than the average purchase price for that product.
5. Find all warehouses that don’t contain a particular product (say Product\_ID = 101).
6. List suppliers who have never been paid (using NOT EXISTS with SupplierPayments).

**🔹 6. Stored Procedures (5)**

1. Create a procedure to insert a new product.
2. Create a procedure to fetch purchase details by purchase ID.
3. Write a procedure to update the product price.
4. Write a procedure to delete a review by customer and product.
5. Create a stored procedure to log inventory action (IN or OUT) given product, warehouse, and quantity.

**🔹 7. Functions (Scalar & Table-Valued) (5)**

1. Create a function to calculate discount price (say 10% off) for any product.
2. Create a function that returns the total stock of a product across all warehouses.
3. Write a function to return average rating of a product.
4. Create a table-valued function that lists all purchases of a given supplier.
5. Write a scalar function that returns the full name and contact of a user by user ID.

**🔹 8. Triggers (5)**

1. Create a trigger that logs inventory changes whenever a row is inserted into Purchases.
2. Write a trigger to prevent deleting products that are used in sales or purchases.
3. Create a trigger to auto-update ProductWarehouseStock after a purchase insert.
4. Trigger to ensure review rating is between 1 and 5.
5. Write a trigger to log failed attempts to insert sales with null price.

**🔹 9. Views (5)**

1. Create a view to display product details with category name and total stock.
2. Create a view to show all purchases with supplier and user information.
3. View that shows review summary per product (avg rating, total reviews).
4. View to show all payments with supplier name and payment status.
5. View that combines inventory log with product and warehouse names.

**🔹 10. Indexes (5)**

1. Create a non-clustered index on Products(Name).
2. Create a unique index on Users(Username).
3. Create an index on Inventory\_Log(action\_date) to speed up reporting.
4. Index the ProductWarehouseStock(product\_id, warehouse\_id) for faster stock queries.
5. Add a filtered index on ProductReviews(rating) where rating < 3.

**11, Dates**

**List all purchases made in the current month.**

**Find the total number of sales made each day in the last 7 days.**

**Show customers who made their first purchase in the last 30 days.**

**Get products that received at least one review in the past 3 months.**

**Find the average purchase price per supplier for purchases made in the year 2024.**

**🗓️ Date-Based Queries (8 questions)**

1. **Find the total number of purchases made each month in 2024.**
2. **List all customers who made purchases only in the last 60 days.**
3. **Show the top 3 most sold products in the current year.**
4. **Find the number of reviews submitted per week over the last 2 months.**
5. **Get suppliers who received at least one payment in the last 90 days.**
6. **Find products added to the system in the last quarter.**
7. **List all sales where the sale date is on a weekend (Saturday or Sunday).**
8. **Show customers who haven’t made any purchase in the last 6 months.**

**🔁 Subquery-Based Queries (7 questions)**

1. **List all products with a price higher than the average price of all products.**  
   *(Scalar subquery)*
2. **Find customers who made more purchases than the average number of purchases per customer.**  
   *(Correlated subquery)*
3. **List suppliers who have supplied more than 5 different products.**  
   *(Using subquery in HAVING clause)*
4. **Show the names of products that have never been sold.**  
   *(Using NOT IN with subquery)*
5. **Find products whose total quantity in stock is greater than all other products.**  
   *(Using ALL in subquery)*
6. **Get warehouses that store at least one product that has never been reviewed.**  
   *(Using EXISTS)*
7. **Find the latest purchase made by each supplier.**  
   *(Using correlated subquery in SELECT clause)*