**Easy (10 Questions)**

1. Write a query to display all columns for all records from the Employees table.
2. Write a query to retrieve the first and last names of employees whose salary exceeds 50,000.
3. Write a query to select all customers from the Customers table ordered alphabetically by CustomerName.
4. Write a query to display the total number of orders from the Orders table.
5. Write a query to display the number of employees in each department.
6. Write a query that selects EmployeeID and concatenates FirstName and LastName as FullName.
7. Write a query to join the Employees and Departments tables to show each employee’s name and their department name.
8. Write a query to calculate and display each product’s price with a 10% discount.
9. Write a query to select all orders where the CustomerID is 'ALFKI', 'ANATR', or 'FRANK'.
10. Write a query to retrieve orders placed after January 1, 2023.

**Intermediate (20 Questions)**

1. Write a query to list the EmployeeID, FirstName, and DepartmentName for all employees.
2. Write a query to show all customers and any matching orders.
3. Write a query to display each department having more than 5 employees.
4. Write a query to select employees whose salary is above the average salary.
5. Write a query to display each department’s name and its total salary expenditure.
6. Write a query to list distinct cities from the Customers table.
7. Write a query to find orders with a TotalAmount between 500 and 2000.
8. Write a query using ROW\_NUMBER() to number orders by OrderDate within each CustomerID.
9. Write a query that selects OrderID and a column OrderStatus that shows 'High' if TotalAmount is above 1000, otherwise 'Low'.
10. Write a query to select employees where DepartmentID is 3 and the Salary is between 40,000 and 80,000.
11. Write a query to select employee names and salaries, ordered by Salary descending then LastName ascending.
12. Write a query to assign a rank to employees based on their salary.
13. Write a query to assign a dense rank to employees within each department based on salary.
14. Write a query to display each order’s OrderDate along with the next order’s OrderDate.
15. Write a query to display each order’s OrderDate along with the previous order’s OrderDate.
16. Write a query to display customers who placed orders with a TotalAmount greater than the average order amount.
17. Write a query to select OrderID, TotalAmount, and the discount using dbo.CalculateDiscount(@TotalAmount).
18. Describe how to call the stored procedure usp\_GetEmployeeDetails with EmployeeID as a parameter.
19. Explain how an index on OrderDate in the Orders table can improve performance.
20. Write a query that shows each order with the customer's name, order date, and employee who handled it.

**Hard (20 Questions)**

1. Write a query to rank employees within each department by salary.
2. Write a query to calculate a running total of TotalAmount for each customer’s orders.
3. Write a query that selects customers from vw\_OrderSummary whose order total is above the average and filters using the Customers table.
4. Explain how to design a stored procedure that returns the top 5 bestselling products.
5. Write a query to list every order with its computed discount using dbo.ComputeCustomDiscount(@OrderID), and filter out discounts below a threshold.
6. Write a query to calculate the median salary for each department.
7. Explain how indexes work and how indexing CustomerID improves join performance.
8. Write a query to compute the rank of each product by total sales using Products, OrderDetails, and Orders.
9. Describe a scenario where an AFTER INSERT trigger updates a sales summary table.
10. Write a query using LAG() to show salary differences and rank employees with the largest jumps.
11. Design a view showing each department with total employees and salary payout. Join it with Departments and filter by payout-to-employee ratio.
12. Write a query using LEAD() and LAG() to compare consecutive TotalAmount values and filter based on dbo.GetThreshold().
13. Explain how a stored procedure can be used within a trigger to update loyalty points.
14. Write a query that uses a nested subquery to calculate average sales per month and ranks months by sales.
15. Write a query to create a customer leaderboard using rank and total purchases.
16. Write a query to calculate year-over-year growth in sales per customer.
17. Write a query that joins Employees with vw\_DepartmentPerformance and ranks departments.
18. Describe a trigger that prevents updates when order status is 'Shipped'.
19. Write a query using both RANK() and DENSE\_RANK() to compare handling of tied values.
20. Design a solution integrating:
    * A view for sales summary
    * A stored procedure to get top-performing regions
    * A UDF for custom commission
    * Indexes for performance
    * A trigger to log orders
    * Window functions for ranking  
      Explain each component’s role, interaction, benefits, and challenges.