SQL Queries interview?

☺ SQL tactics for optimize Query:

- ✓ Join Order in SQL Queries: Starting with a large table may process many rows early, leading to high memory and CPU usage. **Beginning with a smaller table minimizes the initial dataset**, enabling faster joins and reducing overall workload. Also, **Indexing Join Columns**: Utilizing indexes on the columns involved in join conditions can significantly enhance query performance.
- ✓ **Use EXISTS Instead of IN for Subqueries for large datasets**
- ✓ **Use Window Functions Instead of Self-Joins and Correlated Subqueries**
- ✓ **Avoid Leading Wildcards in LIKE Clauses (a LIKE pattern (%value)) instead restructure your data**, you can restructure your data to include additional columns that indicate whether a specific pattern exists, allowing for faster, more efficient queries. By add a Boolean column that flags whether the term appears in the column.
- ✓ **Using Temporary Tables Instead of Common Table Expression CTEs for Large Datasets.** As CTEs are recalculated each time they are referenced, which can be inefficient with large datasets. While Temporary tables are created once, can be indexed, and reused across multiple queries, offering greater efficiency and flexibility.
- ✓ Use Proper Indexing and Know When to Add Indexes, avoid using scalar functions on indexed columns in your WHERE clauses. Every time you use a data table check for the table's indexes in data dictionary and make major joins and group by's on those columns
- ✓ Simplify Complex Logic with **CASE Statements**
- ✓ Avoid **SELECT *** and Specify Columns Explicitly
- ✓ Filter Early with WHERE Clauses
- ✓ Use LIMIT to Restrict Results
- ✓ Eliminate Unnecessary DISTINCT
- ✓ Avoid use function when searching for pattern
- ✓ Avoid use calculated fields in the join and where clauses
- ✓ **Avoid OR Conditions in WHERE Clause use union expect.**

```
SELECT * FROM Employees WHERE Department = 'Sales'
UNION
SELECT * FROM Employees WHERE Department = 'HR';
```

☺ How to read TOP 5 records from a table using a SQL query

```sql
SELECT TOP 5 * FROM table_name;
```

☺ How to read LAST 5 records from a table using a SQL query?

```sql
SELECT TOP 5 * FROM table_name
ORDER BY column_name DESC;
```

☺ How to find the employee with a second MAX Salary using a SQL query?

```sql
SELECT *
FROM employee
WHERE salary = (SELECT MAX(salary)
                FROM employee
                WHERE salary < (SELECT MAX(salary) FROM employee));
```

☺ **How to find the employee with third MAX Salary using a SQL query without using Analytic Functions?**

```sql
SELECT *
FROM employee e1
WHERE 2 = (SELECT COUNT(DISTINCT salary)
           FROM employee e2
           WHERE e2.salary > e1.salary);
```

```sql
SELECT *
FROM employee
ORDER BY salary DESC
LIMIT 1 OFFSET 2;
```

☺ **How would you join two tables on multiple conditions?**

```sql
SELECT a.*, b.*
FROM table1 a
JOIN table2 b
ON a.column1 = b.column1
AND (a.column2 = b.column2 OR a.column3 = b.column3);
```

☺ **Given a table with customer orders, how would you identify the customers who placed the most orders in the last month?**

```sql
SELECT customer_id, COUNT(order_id) AS total_orders
FROM orders
WHERE order_date >= DATEADD(MONTH, -1, GETDATE())  -- Filter for the last month
GROUP BY customer_id
ORDER BY total_orders DESC;
```

☺ **How to delete DUPLICATE records from a table using a SQL Query?**

```sql
WITH CTE AS (
  SELECT *,
    ROW_NUMBER() OVER (PARTITION BY column1, column2, ... ORDER BY column1) AS row_num
  FROM table_name
)
DELETE FROM CTE WHERE row_num > 1;
```

☺ **Write a SQL query to find the second most recent order date for each customer from a table Orders (OrderID, CustomerID, OrderDate).**

```sql
WITH RankedOrders AS (
  SELECT
    CustomerID,
    OrderDate,
    ROW_NUMBER() OVER (PARTITION BY CustomerID ORDER BY OrderDate DESC) AS rank
  FROM Orders
)
SELECT CustomerID, OrderDate
FROM RankedOrders
WHERE rank = 2;
```

☺ **Write a query to find the nth highest salary from a table Employees with columns EmployeeID, Name, and Salary.**

```sql
WITH RankedSalaries AS (
  SELECT
    EmployeeID,
    Name,
    Salary,
    Department,
    DENSE_RANK() OVER (PARTITION BY Department ORDER BY Salary DESC) AS rank
  FROM Employees
)
SELECT EmployeeID, Name, Salary, Department
FROM RankedSalaries
WHERE rank = N; -- Replace N with the desired rank (e.g., 2 for second highest salary)
```

☺ **Given a table Products with columns ProductID, Name, Price, and a table Sales with columns SaleID, ProductID, Quantity, write a query to find the product with the highest revenue.**

```sql
SELECT p.ProductID, p.Name, SUM(p.Price * s.Quantity) AS total_revenue
FROM Products p
JOIN Sales s ON p.ProductID = s.ProductID
GROUP BY p.ProductID, p.Name
ORDER BY total_revenue DESC
LIMIT 1;
```

☺ **Write a query to find the cumulative salary of employees' department-wise, who have joined company in last 30 days.**

```sql
SELECT Department, SUM(Salary) AS CumulativeSalary
FROM Employees
WHERE JoinDate >= CURDATE() - INTERVAL 30 DAY
GROUP BY Department;
```

☺ **Query to find number of employees hired in each quarter of 2023.**

```sql
SELECT
    QUARTER(HireDate) AS Quarter,
    COUNT(EmployeeID) AS HiredEmployees
FROM Employees
WHERE YEAR(HireDate) = 2023
GROUP BY QUARTER(HireDate)
ORDER BY Quarter;
```

☺ **How do you retrieve top 3 departments with highest total salary expenditure?**

```sql
SELECT Department, SUM(Salary) AS TotalSalaryExpenditure
FROM Employees
GROUP BY Department
ORDER BY TotalSalaryExpenditure DESC
LIMIT 3;
```

☺ **How to get unique records without using the DISTINCT keyword.**

```sql
SELECT column1, column2
FROM table_name
GROUP BY column1, column2;
```

☺ **How to select all even or all odd records in a table?**

Selecting even records:

```sql
SELECT *
FROM table_name
WHERE MOD(column_name, 2) = 0;
```

Selecting odd records:

```sql
SELECT *
FROM table_name
WHERE MOD(column_name, 2) <> 0;
```