MOST ASKED

SQL

INTERVIEW QUESTIONS

at MAANG Companies

Medium-Level SQL Questions

Q1.

Explain the difference between INNER JOIN and LEFT JOIN.

Ans.

An **‘INNER JOIN’** returns only the rows where there is a match in both tables. A **‘LEFT JOIN’** returns all rows from the left table and matched rows from the right table, and **‘NULL’** for non-matching rows in the right table.

Q2.

What is the purpose of the GROUP BY clause?

Ans.

The GROUP BY clause groups rows that have the same values in specified columns into summary rows, like "find the number of customers in each country."

Q3.

How can you delete duplicate rows in a SQL table?

Ans.

```sql
DELETE FROM your_table
WHERE id NOT IN (
  SELECT MIN(id)
  FROM your_table
  GROUP BY column_with_duplicates
);
```

Q4.

What are subqueries and how are they used?

Ans.

Subqueries are nested queries used within a main query to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Q5.

Explain the concept of a ‘VIEW’ in SQL.

Ans.

A ‘VIEW’ is a virtual table based on the result set of an SQL query. It can simplify complex queries, improve security by restricting data access, and present a consistent, logical view of the data.

Q6.

What is a ‘UNION’ operator?

Ans.

The ‘UNION’ operator is used to combine the result sets of two or more ‘SELECT’ statements. Each ‘SELECT’ statement must have the same number of columns in the result sets with similar data types.

Q7.

How do you use the ‘CASE’ statement in SQL?

Ans.

sql Copy code

```sql
SELECT name,
  CASE
    WHEN score >= 90 THEN 'A'
    WHEN score >= 80 THEN 'B'
    ELSE 'C'
  END as grade
FROM students;
```

Q8.

What is an ‘INDEX’ and why is it important?

Ans.

An index is a database object that improves the speed of data retrieval operations on a table at the cost of additional writes and storage space.

Q9.

Describe the use of ‘TRIGGERS’ in SQL.

Ans.

Triggers are database objects that automatically execute a specified SQL procedure when a certain event occurs on a table or view, such as an ‘INSERT’, ‘UPDATE’, or ‘DELETE’.

Q10.

Explain the concept of ‘NORMALIZATION’.

Ans.

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller ones and defining relationships between them.

Advanced SQL Questions

Q11.

What is a ‘CTE’ (Common Table Expression)?

Ans.

sql Copy code

```sql
WITH Sales_CTE (SalesPerson, TotalSales) AS (
  SELECT SalesPerson, SUM(SalesAmount)
  FROM Sales
  GROUP BY SalesPerson
)
SELECT * FROM Sales_CTE;
```

Q12.

Explain the differences between ‘DELETE’, ‘TRUNCATE’, and ‘DROP’ commands.

Ans.

‘DELETE’: Removes rows one at a time and logs individual row deletions. It can include a WHERE clause.

‘TRUNCATE’: Removes all rows from a table without logging individual row deletions. Cannot include a ‘WHERE’ clause.

‘DROP’: Removes a table from the database entirely, including its structure.

Q13.

How do you optimize a slow-running query?

Ans.

Techniques include indexing, query rewriting, reducing the number of joins, avoiding SELECT *, and analyzing the execution plan.

Q14.

What are ‘window functions’ in SQL?

Ans.

sql Copy code

```sql
SELECT name, department,
       salary,
       RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rank
FROM employees;
```

Q15.

Discuss the concept of ‘ACID’ properties in transactions.

Ans.

Atomicity: Ensures that all operations within a transaction are completed; if not, the transaction is aborted.

Consistency: Ensures that a transaction brings the database from one valid state to another.

Isolation: Ensures that transactions are securely and independently processed at the same time without interference.

Durability: Ensures that once a transaction is committed, it will remain so, even in the event of a power loss, crash, or error.

Q16.

How do you handle transactions in SQL?

Ans.

```sql
BEGIN TRANSACTION;
-- SQL statements
COMMIT;
```

Q17.

What is the difference between ‘HAVING’ and ‘WHERE’ clauses?

Ans.

‘WHERE’ is used to filter records before any groupings are made, while ‘HAVING’ is used to filter values after grouping.

Q18.

Explain the concept of ‘sharding’ in databases.

Ans.

Sharding is a type of database partitioning that splits large databases into smaller, more manageable pieces, called shards, which can be spread across multiple servers.

Q19.

What are ‘stored procedures’ and how are they used?

Ans.

Stored procedures are precompiled collections of SQL statements stored under a name and processed as a unit. They can accept parameters and are used to perform tasks like data validation or access control.

Q20.

Explain the concept of ‘Data Warehousing’.

Ans.

Data warehousing involves the collection, storage, and management of large volumes of data from multiple sources to provide meaningful business insights. It typically involves ETL processes and uses schemas like star or snowflake schema.

Scenario-Based SQL Questions

Scenario 1:

Employee Management System

Tables:

- Employees

```sql
CREATE TABLE Employees (
  EmployeeID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  DepartmentID INT,
  Salary DECIMAL(10, 2),
  HireDate DATE
);
```

- Departments

```sql
CREATE TABLE Departments (
  DepartmentID INT PRIMARY KEY,
  DepartmentName VARCHAR(50)
);
```

Questions:

Q21.

Find the second highest salary in the Employees table.

Ans.

```sql
SELECT MAX(Salary) AS SecondHighestSalary
FROM Employees
WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

Q22.

List all employees who have been hired in the last year.

Ans.

```sql
SELECT * FROM Employees
WHERE HireDate >= DATEADD(year, -1, GETDATE());
```

Q23.

Find the average salary for each department.

Ans.

sql Copy code

```sql
SELECT d.DepartmentName, AVG(e.Salary) AS AverageSalary
FROM Employees e
JOIN Departments d ON e.DepartmentID = d.DepartmentID
GROUP BY d.DepartmentName;
```

Q24.

List all departments along with the count of employees in each department.

Ans.

sql Copy code

```sql
SELECT d.DepartmentName, COUNT(e.EmployeeID) AS EmployeeCount
FROM Departments d
LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID
GROUP BY d.DepartmentName;
```

Scenario 2:

Online Retail Database

Tables:

- Orders

```sql
CREATE TABLE Orders (
  OrderID INT PRIMARY KEY,
  CustomerID INT,
  OrderDate DATE,
  TotalAmount DECIMAL(10, 2)
);
```

- OrderDetails

```sql
CREATE TABLE OrderDetails (
  OrderDetailID INT PRIMARY KEY,
  OrderID INT,
  ProductID INT,
  Quantity INT,
  Price DECIMAL(10, 2)
);
```

- Products

sql Copy code

```sql
CREATE TABLE Products (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(100),
  CategoryID INT,
  Price DECIMAL(10, 2)
);
```

Questions:

Q25.

Find the top 3 most sold products in the last month.

Ans.

sql Copy code

```sql
SELECT TOP 3 p.ProductName, SUM(od.Quantity) AS TotalSold
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
JOIN Orders o ON od.OrderID = o.OrderID
WHERE o.OrderDate >= DATEADD(month, -1, GETDATE())
GROUP BY p.ProductName
ORDER BY TotalSold DESC;
```

Q26.

Calculate the total sales for each product category.

Ans.

```sql
SELECT c.CategoryName, SUM(od.Quantity * od.Price) AS TotalSales
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
JOIN Categories c ON p.CategoryID = c.CategoryID
GROUP BY c.CategoryName;
```

Q27.

Find all customers who placed orders in the last 7 days.

Ans.

```sql
SELECT DISTINCT CustomerID
FROM Orders
WHERE OrderDate >= DATEADD(day, -7, GETDATE());
```

Q28.

Calculate the average order value for each customer.

Ans.

```sql
SELECT CustomerID, AVG(TotalAmount) AS AverageOrderValue
FROM Orders
GROUP BY CustomerID;
```

Q29.

Identify the product that generated the highest revenue.

Ans.

```sql
SELECT p.ProductName, SUM(od.Quantity * od.Price) AS TotalRevenue
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY TotalRevenue DESC
LIMIT 1;
```

Q30.

List customers who have placed more than 5 orders.

Ans.

```sql
SELECT CustomerID, COUNT(OrderID) AS OrderCount
FROM Orders
GROUP BY CustomerID
HAVING COUNT(OrderID) > 5;
```

Q31.

Find the month-over-month growth rate of total sales.

Ans.

```sql
SELECT
    DATE_FORMAT(OrderDate, '%Y-%m') AS Month,
    SUM(TotalAmount) AS TotalSales,
    (SUM(TotalAmount) - LAG(SUM(TotalAmount), 1) OVER (ORDER BY DATE_FORMAT
    (OrderDate, '%Y-%m'))) / LAG(SUM(TotalAmount), 1) OVER (ORDER BY DATE_FORMAT
    (OrderDate, '%Y-%m')) * 100 AS GrowthRate
FROM Orders
GROUP BY DATE_FORMAT(OrderDate, '%Y-%m');
```

Q32.

Identify the employees who processed the highest number of orders.

Ans.

```sql
SELECT e.EmployeeID, COUNT(o.OrderID) AS OrderCount
FROM Orders o
JOIN Employees e ON o.EmployeeID = e.EmployeeID
GROUP BY e.EmployeeID
ORDER BY OrderCount DESC
LIMIT 1;
```

Q33.

Find the products that have not been sold in the last 6 months.

Ans.

```sql
SELECT p.ProductName
FROM Products p
LEFT JOIN OrderDetails od ON p.ProductID = od.ProductID
LEFT JOIN Orders o ON od.OrderID = o.OrderID AND o.OrderDate >= DATEADD
(month, -6, GETDATE())
WHERE o.OrderID IS NULL;
```

Q34.

List customers and the total amount they have spent.

Ans.

```sql
SELECT CustomerID, SUM(TotalAmount) AS TotalSpent
FROM Orders
GROUP BY CustomerID;
```

Q35.

Calculate the average number of items per order.

Ans.

```sql
SELECT AVG(ItemCount) AS AvgItemsPerOrder
FROM (
    SELECT OrderID, SUM(Quantity) AS ItemCount
    FROM OrderDetails
    GROUP BY OrderID
) AS OrderItemCounts;
```

Q36.

Find overlapping date ranges for product availability.

Ans.

sql Copy code

```sql
CREATE TABLE ProductAvailability (
    ProductID INT,
    StartDate DATE,
    EndDate DATE
);

SELECT a.ProductID, a.StartDate, a.EndDate, b.StartDate, b.EndDate
FROM ProductAvailability a, ProductAvailability b
WHERE a.ProductID = b.ProductID
AND a.StartDate < b.EndDate
AND b.StartDate < a.EndDate
AND a.StartDate <> b.StartDate;
```

Q37.

Retrieve the most recent order for each customer.

Ans.

```sql
SELECT CustomerID, MAX(OrderDate) AS MostRecentOrderDate
FROM Orders
GROUP BY CustomerID;
```

Q38.

List products that have been ordered more than 100 times.

Ans.

```sql
SELECT p.ProductName, SUM(od.Quantity) AS TotalOrdered
FROM Products p
JOIN OrderDetails od ON p.ProductID = od.ProductID
GROUP BY p.ProductName
HAVING SUM(od.Quantity) > 100;
```

Q39.

Create a query to find the longest consecutive sequence of orders for each customer.

Ans.

sql Copy code

```sql
WITH ConsecutiveOrders AS (
    SELECT
        CustomerID,
        OrderDate,
        ROW_NUMBER() OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS RowNum
    FROM Orders
)
SELECT CustomerID, COUNT(*) AS LongestStreak
FROM ConsecutiveOrders
GROUP BY CustomerID, DATE_SUB(OrderDate, INTERVAL RowNum DAY)
ORDER BY LongestStreak DESC;
```

Q40.

Fetch the last N records in a table, ordered by a specific column.

Ans.

sql Copy code

```sql
SELECT *
FROM Orders
ORDER BY OrderDate DESC
LIMIT N;
```

Q41.

Audit data changes in a SQL database.

Ans.

sql Copy code

```sql
CREATE TABLE AuditLog (
    AuditID INT PRIMARY KEY,
    TableName VARCHAR(50),
    Action VARCHAR(10),
    OldValue VARCHAR(255),
    NewValue VARCHAR(255),
    ChangeDate DATETIME,
    ChangedBy VARCHAR(50)
);

CREATE TRIGGER trgAudit
AFTER UPDATE ON Orders
FOR EACH ROW
BEGIN
    INSERT INTO AuditLog (TableName, Action, OldValue, NewValue, ChangeDate,
                          ChangedBy)
    VALUES ('Orders', 'UPDATE', OLD.TotalAmount, NEW.TotalAmount, NOW(), USER());
END;
```

Q42.

Design a query to retrieve hierarchical data in a parent-child relationship.

Ans.

sql Copy code

```sql
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(50),
    ManagerID INT
);

WITH EmployeeHierarchy AS (
    SELECT EmployeeID, Name, ManagerID, 1 AS Level
    FROM Employees
    WHERE ManagerID IS NULL
    UNION ALL
    SELECT e.EmployeeID, e.Name, e.ManagerID, eh.Level + 1
    FROM Employees e
    JOIN EmployeeHierarchy eh ON e.ManagerID = eh.EmployeeID
)
SELECT * FROM EmployeeHierarchy
ORDER BY Level, ManagerID;
```

Q43.

Implement soft deletes in SQL.

Ans.

sql — Copy code

```sql
ALTER TABLE Orders ADD COLUMN IsDeleted BOOLEAN DEFAULT FALSE;

-- Mark as deleted
UPDATE Orders SET IsDeleted = TRUE WHERE OrderID = 1;

-- Retrieve non-deleted orders
SELECT * FROM Orders WHERE IsDeleted = FALSE;
```

Q44.

Write a query to calculate the year-over-year growth of sales.

Ans.

sql — Copy code

```sql
SELECT
    YEAR(OrderDate) AS Year,
    SUM(TotalAmount) AS TotalSales,
    (SUM(TotalAmount) - LAG(SUM(TotalAmount), 1) OVER (ORDER BY YEAR(OrderDate)))
    / LAG(SUM(TotalAmount), 1) OVER (ORDER BY YEAR(OrderDate)) * 100 AS YoYGrowth
FROM Orders
GROUP BY YEAR(OrderDate);
```

Q45.

Handle data migration from one database to another.

Ans.

```sql
-- Using MySQL Workbench or similar tools for data migration.
-- Example:
mysqldump -u source_user -p source_db > source_db.sql
mysql -u target_user -p target_db < source_db.sql
```

Q46.

How to implement full-text search in SQL?

Ans.

```sql
CREATE FULLTEXT INDEX idx_productname ON Products(ProductName);

SELECT * FROM Products
WHERE MATCH(ProductName) AGAINST('search term');
```

Q47.

Write a query to find the median value in a numeric column.

Ans.

```sql
SELECT AVG(value) AS MedianValue
FROM (
    SELECT value
    FROM table
    ORDER BY value
    LIMIT 2 - (SELECT COUNT(*) FROM table) % 2
    OFFSET (SELECT (COUNT(*) - 1) / 2 FROM table)
) AS MedianValues;
```

Q48.

How to perform a bulk insert in SQL?

Ans.

```sql
INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
VALUES
(1, 1, '2023-01-01', 100.00),
(2, 2, '2023-01-02', 150.00),
(3, 1, '2023-01-03', 200.00);
```

Q49.

Calculate the running total of sales for each day.

Ans.

```sql
SELECT OrderDate,
       TotalAmount,
       SUM(TotalAmount) OVER (ORDER BY OrderDate) AS RunningTotal
FROM Orders;
```

Q50.

Identify customers who have not placed any orders in the last 6 months.

Ans.

```sql
SELECT CustomerID
FROM Customers
WHERE CustomerID NOT IN (
    SELECT DISTINCT CustomerID
    FROM Orders
    WHERE OrderDate >= DATEADD(month, -6, GETDATE())
);
```