Part A – Conceptual Questions

SQL Basics

1. What is SQL and how is it different from MySQL/SQL Server/Oracle?
2. Explain **DDL, DML, DCL, TCL** with examples.
3. What are **constraints** in SQL? Name all types.
4. What is the difference between **PRIMARY KEY** and **UNIQUE KEY**?
5. What is the difference between **DELETE**, **TRUNCATE**, and **DROP**?
6. Explain the difference between **CHAR** and **VARCHAR**.
7. What is the difference between **WHERE** and **HAVING**?
8. What are **aggregate functions** in SQL? Give examples.
9. What is the difference between **COUNT(*)** and **COUNT(column_name)**?
10. Explain the difference between **BETWEEN** and **IN**.

Joins

11. What is the difference between **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**, and **FULL JOIN**?
12. What is a **SELF JOIN**?
13. What are **CROSS JOIN** and **CARTESIAN PRODUCT**?
14. What is the difference between **ANTI JOIN** and **SEMI JOIN**?
15. Can we join a table to itself? How?

Indexes & Performance

16. What is an **index** in SQL? Why is it used?

17. Difference between **Clustered** and **Non-Clustered Index**.
18. What is a **composite index**?
19. Does an index improve `INSERT` performance? Why/Why not?
20. What is an **execution plan** in SQL?

Advanced Concepts

21. What is a **subquery**? Difference between **correlated** and **non-correlated** subqueries.
22. What is the difference between **View** and **Materialized View**?
23. Can we perform **DML** operations on a view?
24. What is a **stored procedure**?
25. Difference between **stored procedure** and **function**.
26. What are **window functions** in SQL? Name a few.
27. Difference between **RANK()**, **DENSE_RANK()**, and **ROW_NUMBER()**.
28. What is the difference between **LEAD** and **LAG** functions?
29. What are **CTEs** in SQL? How are they different from subqueries?
30. What is the difference between **UNION** and **UNION ALL**?

Transactions

31. What are **ACID properties**?
32. Difference between **COMMIT** and **ROLLBACK**.
33. What is a **SAVEPOINT** in SQL?
34. Difference between **Implicit** and **Explicit** transactions.

Data Engineering Focus

35. How do you handle **NULL** values in SQL?
36. Difference between `IS NULL` and `= NULL`.
37. How can you find duplicate records in a table?
38. How to remove duplicates while keeping only one record?
39. What is **normalization**? Name different normal forms.
40. Difference between **OLTP** and **OLAP** databases.

Part B – Practical Questions

(Students must write and run queries on a given dataset)

Basic Queries

41. Create a table `Customers` with columns: `CustomerID`, `Name`, `City`, `Country`. Add appropriate constraints.
42. Insert 5 sample rows into `Customers`.
43. Write a query to fetch all customers from `India`.
44. Write a query to fetch customers whose name starts with 'A'.
45. Write a query to fetch customers whose name ends with 'n'.

Aggregations

46. Write a query to find the total number of customers in each country.
47. Write a query to find the highest and lowest `CustomerID`.
48. Write a query to count how many customers are from each city.

Joins

49. Create another table `Orders` with columns: `OrderID`, `CustomerID`, `OrderDate`, `Amount`.

50. Insert 5–6 sample orders for different customers.

51. Write an **INNER JOIN** to fetch orders along with customer names.

52. Write a **LEFT JOIN** to fetch all customers and their orders.

53. Write a **RIGHT JOIN** to fetch all orders with customer details.

54. Write a **SELF JOIN** on `Customers` to find customers from the same city.

Advanced Queries

55. Write a query to find the second highest order amount.

56. Write a query to rank customers based on total spending.

57. Write a query to get the first and last order date for each customer.

58. Write a query to calculate the running total of order amounts.

59. Write a query to find customers who placed more than 2 orders.

60. Write a query to display customers who never placed an order.

Subqueries & CTEs

61. Write a query to fetch customers whose total order amount is greater than the average order amount.

62. Write a **correlated subquery** to fetch customers who placed orders worth more than 500.

63. Use a **CTE** to calculate each customer's total order amount and rank them.

Set Operations

64. Create another table `VIP_Customers` with a list of special customers.

65. Write a query to get all customers from both `Customers` and `VIP_Customers` using **UNION**.
66. Write a query to get customers present in both tables (**INTERSECT** equivalent).
67. Write a query to get customers in `Customers` but not in `VIP_Customers`.

Indexes & Optimization

68. Create an index on `CustomerID` in `Orders`.
69. Check the query execution plan for a JOIN query with and without the index.

Transactions

70. Begin a transaction, insert a record, rollback, and verify if it is saved.
71. Begin a transaction, insert a record, commit, and verify if it is saved.

NULL Handling

72. Write a query to replace NULL amounts in `Orders` with 0.
73. Write a query to count orders where `Amount` is NULL.

Date Functions

74. Write a query to fetch orders placed in the last 30 days.
75. Write a query to fetch orders placed in January 2024.

String Functions

76. Write a query to convert all customer names to UPPERCASE.
77. Write a query to extract the first 3 letters of customer names.
78. Write a query to concatenate `Name` and `City` with a comma.

Miscellaneous

79. Delete all customers from a specific city.

80. Drop the `VIP_Customers` table.

Part C – Simple Scenario-Based SQL Questions

Scenario 1 – Employee Database

Tables:

```
Employees(emp_id, emp_name, dept_id, salary, joining_date)
Departments(dept_id, dept_name, location)
```

1. Write a query to fetch all employees working in the `IT` department.
2. Write a query to fetch the top 3 highest-paid employees in the company.
3. Find the total salary paid to employees in each department.
4. List departments where the average salary is greater than 50,000.
5. Find employees who joined in the year 2023.
6. Fetch employees whose salary is above the company's average salary.
7. List employees who do not belong to any department.

Scenario 2 – E-commerce Orders

Tables:

```
Orders(order_id, customer_id, order_date, amount)
Customers(customer_id, customer_name, city)
```

8. List all orders placed by customers from `Mumbai`.
9. Find the total amount spent by each customer.
10. Find customers who have not placed any orders.

11. List the top 2 customers based on total spending.
12. Find orders placed in the last 90 days.
13. Find the highest single order amount for each customer.

Scenario 3 – School Database

Tables:

```
Students(student_id, student_name, class_id, marks)
Classes(class_id, class_name)
```

14. List all students in `Class 10`.
15. Find students who scored more than 80 marks.
16. Find the average marks for each class.
17. List students who scored above the class average.
18. Find students who are not assigned to any class.

Scenario 4 – Banking Transactions

Tables:

```
Accounts(account_id, account_holder, balance)
Transactions(txn_id, account_id, txn_date, amount, txn_type)
```

(*txn_type = 'credit' or 'debit'*)

19. Fetch all transactions for account ID `1001`.
20. Find the total credited amount for each account.
21. List accounts that have never made a transaction.
22. Find accounts where the balance is less than 1000.

23. Fetch the last transaction date for each account.

24. Calculate the net transaction amount (credits – debits) for each account.

Scenario 5 – Movie Database

Tables:

```
Movies(movie_id, title, release_year, rating)
Actors(actor_id, actor_name)
Movie_Actors(movie_id, actor_id)
```

25. List all movies released in 2022.

26. Find movies with a rating above 8.5.

27. List all actors who worked in the movie `Inception`.

28. Count the number of movies each actor has acted in.

29. Find movies that have more than 3 actors.