Assignment - 3 (Interview Questions)

**(1) Database Concepts**

**1. What is a database?**  
A database is an organized collection of structured data stored electronically and managed by a DBMS.

**2. Difference between a database and a DBMS:**

* **Database:** Actual data storage (tables, records).
* **DBMS:** Software used to create, manage, and access the database.

**3. Types of databases:**  
Relational, NoSQL, Object-oriented, Hierarchical, Network, Distributed, and Cloud databases.

**4. What is a relational database?**  
A database that stores data in tables (rows & columns) and uses **relationships** between tables via keys.

**5. What is normalization? Explain its types.**  
Normalization organizes data to reduce redundancy.  
Types:

* **1NF:** Remove repeating groups.
* **2NF:** Remove partial dependencies.
* **3NF:** Remove transitive dependencies.
* **BCNF:** Every determinant is a candidate key.

**6. What is denormalization?**  
Process of combining normalized tables to improve query performance (opposite of normalization).

**7. What is a primary key? Difference from unique key:**

* **Primary key:** Uniquely identifies each record, **cannot be NULL**.
* **Unique key:** Also ensures uniqueness but **can contain NULL**.

**8. What is a foreign key?**  
A key in one table that refers to the primary key in another table, maintaining **referential integrity**.

**9. What are indexes? Why are they used?**  
Indexes are data structures that improve **query performance** by allowing faster data retrieval.

**10. What is a composite key?**  
A key made up of **two or more columns** used together to uniquely identify a record.

**(2) MySQL Commands**

**11. CREATE command:**  
Used to create databases, tables, views, etc.  
CREATE DATABASE db\_name;

**12. Delete a database:**  
DROP DATABASE db\_name;

**13. ALTER command:**  
Used to **modify** an existing table (add, delete, or modify columns).  
ALTER TABLE table\_name ADD column\_name datatype;

**14. Create a table:**

CREATE TABLE students (

id INT PRIMARY KEY,

name VARCHAR(50),

age INT

);

**15. DROP command:**  
Deletes a table or database permanently.  
DROP TABLE table\_name;

**16. Insert data:**  
INSERT INTO students (id, name, age) VALUES (1, 'Adnan', 22);

**17. Update syntax:**  
UPDATE students SET age = 23 WHERE id = 1;

**18. Delete records:**  
DELETE FROM students WHERE id = 1;

**19. SELECT statement:**  
Used to retrieve data.  
SELECT \* FROM students;

**20. Retrieve unique records:**  
SELECT DISTINCT column\_name FROM table\_name;

**(3) Clauses and Operators**

**21. WHERE clause:**  
Filters records based on a condition.  
SELECT \* FROM students WHERE age > 20;

**22. ORDER BY clause:**  
Sorts result in ascending or descending order.  
ORDER BY name ASC;

**23. GROUP BY clause:**  
Groups rows sharing a property.  
SELECT dept, COUNT(\*) FROM employees GROUP BY dept;

**24. HAVING clause:**  
Filters groups after aggregation.  
HAVING COUNT(\*) > 2;

**25. Comparison operators:**  
=, !=, <, >, <=, >=

**26. BETWEEN operator:**  
Filters data within a range.  
WHERE age BETWEEN 18 AND 25;

**27. LIKE operator:**  
Searches for a pattern.  
WHERE name LIKE 'A%';

**28. IN operator:**  
Matches values from a list.  
WHERE dept IN ('IT', 'HR');

**29. NULL operator:**  
Checks for null values.  
WHERE salary IS NULL;

**30. AND vs OR:**

* **AND:** All conditions must be true.
* **OR:** Any one condition must be true.

**(4) Predefined Functions**

**31. Aggregate functions:**  
Perform calculations on multiple values — e.g., SUM(), AVG(), COUNT(), MAX(), MIN().

**32. COUNT():**  
Returns number of records.  
SELECT COUNT(\*) FROM employees;

**33. SUM():**  
Adds numeric column values.  
SELECT SUM(salary) FROM employees;

**34. AVG():**  
Calculates average.  
SELECT AVG(salary) FROM employees;

**35. MAX():**  
Finds highest value.  
SELECT MAX(salary) FROM employees;

**36. MIN():**  
Finds lowest value.  
SELECT MIN(salary) FROM employees;

**37. String functions:**  
LENGTH(), UPPER(), LOWER(), CONCAT(), SUBSTRING()

**38. CONCAT():**  
Joins strings.  
SELECT CONCAT(first\_name, ' ', last\_name);

**39. SUBSTRING():**  
Extracts part of a string.  
SUBSTRING(name, 1, 3);

**40. NOW():**  
Returns current date and time.

**(5) User-Defined Functions**

**41. What is UDF?**  
A custom function created by users to perform specific operations.

**42. Create a UDF:**

CREATE FUNCTION get\_bonus(salary DECIMAL(10,2))

RETURNS DECIMAL(10,2)

RETURN salary \* 0.1;

**43. Call UDF:**  
SELECT get\_bonus(50000);

**44. Can UDFs return multiple values?**  
No — a UDF returns **one value only**.

**45. Advantages of UDFs:**  
Code reuse, modularity, readability, and maintainability.

**(6) Views**

**46. What is a view?**  
A **virtual table** based on the result of a query.

**47. Create a view:**  
CREATE VIEW high\_salary AS SELECT name, salary FROM employees WHERE salary > 50000;

**48. View vs table:**

* View: Virtual, no physical storage.
* Table: Physically stores data.

**49. Can you update a view?**  
Yes, if it’s based on a single table and doesn’t use joins or aggregates.

**50. Drop a view:**  
DROP VIEW view\_name;

**(7) Common Table Expressions (CTE)**

**51. What is a CTE?**  
A temporary result set defined using WITH that can be referenced in a query.

**52. Create a CTE:**

WITH cte AS (

SELECT dept, COUNT(\*) AS emp\_count FROM employees GROUP BY dept

)

SELECT \* FROM cte;

**53. CTE vs subquery:**  
CTE is reusable and more readable; subquery is nested and temporary.

**54. Recursive CTE:**  
Yes, CTEs can be recursive — useful for hierarchical data.

**55. Reference a CTE:**  
By using its name in the main query (SELECT \* FROM cte;).

**(8) Joins**

**56. What is a join?**  
Combines rows from two or more tables based on related columns.

**57. Types of joins:**  
INNER, LEFT, RIGHT, FULL, CROSS, SELF.

**58. INNER JOIN:**  
Returns matching rows.  
SELECT \* FROM emp e INNER JOIN dept d ON e.dept\_id = d.dept\_id;

**59. LEFT JOIN:**  
Returns all from left + matched from right.

**60. RIGHT JOIN:**  
Returns all from right + matched from left.

**61. FULL OUTER JOIN:**  
Returns all records from both tables (not directly supported in MySQL — use UNION).

**62. CROSS JOIN:**  
Cartesian product — every row from table A with every row from table B.

**63. Self-join:**  
A table joined with itself.

**64. Join multiple tables:**  
Use multiple JOIN clauses in a single query.

**65. Join vs subquery:**  
Join merges data; subquery returns results used by the outer query.

**(9) Subqueries**

**66. What is a subquery?**  
A query inside another query.

**67. Subquery in SELECT:**  
SELECT name, (SELECT dept\_name FROM dept WHERE dept\_id = e.dept\_id) FROM emp e;

**68. Subquery in WHERE:**  
SELECT \* FROM emp WHERE salary > (SELECT AVG(salary) FROM emp);

**69. Correlated subquery:**  
References columns from outer query; executes once per row.

**70. Handle multiple rows:**  
Use operators like IN, ANY, ALL.

**(10) Stored Procedures**

**71. What is a stored procedure?**  
A precompiled SQL code block stored in the database.

**72. Create a stored procedure:**

DELIMITER //

CREATE PROCEDURE getAllEmployees()

BEGIN

SELECT \* FROM employees;

END //

DELIMITER ;

**73. Call a procedure:**  
CALL getAllEmployees();

**74. Accept parameters:**  
Yes, using IN, OUT, INOUT.

**75. Advantages:**  
Improves performance, reusability, and security.

**(11) Triggers**

**76. What is a trigger?**  
A block of SQL that automatically executes on events like INSERT, UPDATE, DELETE.

**77. Create a trigger:**

CREATE TRIGGER before\_insert

BEFORE INSERT ON employees

FOR EACH ROW

SET NEW.created\_at = NOW();

**78. Types of triggers:**  
BEFORE and AFTER (INSERT, UPDATE, DELETE).

**79. Can a trigger call a stored procedure?**  
Yes.

**80. Difference between trigger and stored procedure:**

* Trigger: Executes automatically.
* Procedure: Executed manually.

**(12) Data Control Language (DCL)**

**81. What is DCL?**  
Manages database permissions and access control.

**82. GRANT:**  
Gives privileges.  
GRANT SELECT, INSERT ON db.table TO 'user'@'localhost';

**83. REVOKE:**  
Removes privileges.  
REVOKE SELECT ON db.table FROM 'user'@'localhost';

**84. User vs role:**  
User = individual account; Role = group of privileges.

**85. Create a user:**  
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';

**(13) Transaction Control Language (TCL)**

**86. What is TCL?**  
Manages transactions to ensure data consistency.

**87. COMMIT:**  
Saves all changes permanently.

**88. ROLLBACK:**  
Undoes changes since last COMMIT.

**89. SAVEPOINT:**  
Sets a point to rollback to partially.

**90. Set isolation level:**  
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

**(14) Types of Databases**

**91. Types:**  
Relational, NoSQL, Distributed, Cloud, Graph, Object-oriented.

**92. SQL vs NoSQL:**

* SQL: Structured, table-based, uses schema.
* NoSQL: Unstructured/semi-structured, schema-less.

**93. Examples of NoSQL:**  
MongoDB, Cassandra, Redis, CouchDB, DynamoDB.

**94. Distributed database:**  
Data stored across multiple servers.

**95. Cloud database:**  
Hosted on cloud platforms (e.g., AWS RDS, Azure SQL).

**(15) Database Management Systems (DBMS)**

**96. What is DBMS?**  
Software for storing, managing, and retrieving data.

**97. Functions:**  
Data storage, retrieval, backup, security, and concurrency control.

**98. DBMS vs RDBMS:**  
RDBMS supports relations and normalization; DBMS does not.

**99. Popular DBMS:**  
MySQL, PostgreSQL, Oracle, SQL Server, MongoDB.

**100. Data integrity:**  
Accuracy & consistency of data; ensured via constraints (PK, FK, CHECK, etc.).