SAFFRON | Page No.
Date / /

1.) What is a database?
→ A database is an organized collection of structured data that can be easily accessed, managed and updated.
Example:-
A library database stores details like:
- Book_ID : lol
- Title : "The Alchemist"
- Author : "Paulo Coelho"
- Status : "Available"

2.) What is the difference between a database and a DBMS?

→ Feature	Database	DBMS
• Definition	A collection of data	Software used to manage and interacts with the database.
• Function	Just stores the data	Allows operations like insert, update, delete
• Example	MySQL data files	MySQL, oracle, PostgreSQL

3.) What are the different types of databases?
1.) Relational Databases (RDMS):- Tables with rows and columns.
Example:- MySQL, PostgreSQL

2.) NoSQL Databases:- Non-tabular, for unstructured data
Example:- MongoDB

3.) Distributed Databases:- Data spread across multiple locations.
Example:- Apache Cassandra.

4.) Cloud Databases:- Hosted on cloud services.
Example:- Amazon RDS

5.) Object-oriented Databases:- Store data as objects
Example:- db4o

4.) What is relational database?
→ A relational database stores data in tables (relations) where each table has rows and columns, and relationships can be established using keys.

5.) What is normalization? Explain its types.
→ Normalization is the process of organizing data to reduce redundancy and improve data integrity.
Types:-
1NF (First Normal form): No repeating groups
2NF (second Normal form): 1NF + No partial dependancy
3NF (Third Normal form): 2NF + No transitive dependancy.

6.) What is denormalization?
→ Denormalization is the process of adding redundancy to a database (reversing normalization) to improve read performance at the cost of write efficiency.

7.) What is a primary key? How is it different from a unique key?
→ Primary key :- Uniquely identifies each row in a table. Cannot be NULL.
Unique key :- Ensures values are unique in a column but can contain NULL.

Examples :-
Student_ID : Primary key
Email : Unique key (can be NULL for some students)

8.) What is a foreign key?
→ A foreign key is a field in one table that refers to the primary key in another table. It establishes a relationship between two tables.

9.) What are indexes? Why are they used?
→ An index is a data structure that improves the speed of data retrival operations on a database table at the cost of additional space.
Example:-
If you're searching for a student by email then an index on Email column speeds up that query:
CREATE INDEX idx_email ON student(Email);

10) What is a Composite key?
→ A composite key is a combination of two a more columns used together as a primary key.
Example :-

```
Create table CourseRegistrations (
  StudentID INT,
  CourseID INT,
  PRIMARY KEY (STUDENTID, COURSE_ID)
);
```

Here, both StudentID and Course_ID together form the primary key (composite key).

11) What is the purpose of the CREATE command?
→ The CREATE command is used to create new databases, tables, view or other database objects.
Example :-

```
Create database college;

Create table students (
  student_ID INT PRIMARY KEY,
  Name VARCHAR(50),
  Age INT
);
```

12) How do you delete a database in MySQL.
→ Use the DROP DATABASE command to permanently delete a database.
Example :-

```
DROP DATABASE College;
```

13.) What is the ALTER command used for?
→ The ALTER command is used to modify the structure of an existing table, such as adding, deleting or modifying columns.

Ex:- ALTER TABLE Students
ADD Email varchar(100);

14.) How do you create a table in MySQL?
→ Use the CREATE TABLE command with column definitions.
Ex:-
CREATE TABLE Employees (
Emp_ID INT PRIMARY KEY,
Name VARCHAR(100)
);

15.) What is the DROP command?
→ The DROP command is used to delete tables, databases or other objects permanently.
Example:
DROP TABLE Employees;

16.) How do you insert data into a table?
→ Use the INSERT INTO statement.
Ex:-
INSERT INTO Employees (Emp_ID, Name, Department, Salary)
VALUES (101, 'Rahul', 'HR', 35000.00);

17) What is the syntax for updating records in a table?

→ Use the UPDATE statement with SET and WHERE clause.

Ex:-

```
UPDATE Employees
SET Salary = 60000.00
WHERE Emp_ID = 101;
```

18) How do you delete records from a table?

→ Use the Delete from statement with a WHERE clause.

Ex:-

```
DELETE FROM Employees
WHERE Emp_ID = 101;
```

19) What is the SELECT statement used for?

→ The SELECT statement is used ~~for~~ to retrieve data from one or more tables.

Ex:-

```
SELECT * FROM Employees;
```

20) How do you retrieve unique records from a table?

→ Use the SELECT DISTINCT keyword to eliminate duplicate rows.

Ex:-

```
SELECT DISTINCT department FROM Employees;
```

21) What is the purpose of the WHERE clause?
→ The WHERE clause is used to filter records that meet a specific condition.

Ex:-
```
SELECT * FROM Employees
WHERE Department = 'HR';
```

22) Explain the ORDER BY clause.
→ The ORDER BY clause is used to sort the results sets by one or more columns.

ex:-
```
SELECT * FROM employees
ORDER BY salary ASC;
```

23) What is the GROUP BY clause used for?
→ The GROUP BY clause is used to group rows that have the same values in specified columns often used with aggregate functions.

ex:-
```
SELECT Departments, COUNT(*) AS total_Employees
FROM Employees
GROUP BY Department;
```

24) How do you use the HAVING clause?
→ HAVING is used to filter grouped data, like a WHERE clause for groups.

Ex:-
```
SELECT Department, COUNT(*) AS Total
FROM Employees
GROUP BY Department
HAVING COUNT(*) > 2;
```

25) What are the different comparison operators in MySQL?

→

Operator	Description
=	Equal to
!= or <>	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal
<=	less than or equal.

26) What is the BETWEEN operator?

→ The BETWEEN operator checks if a value is within a range.

ex:-
```
Select * from Employees
WHERE Salary BETWEEN 30000 AND 50000;
```

27) Explain the LIKE operator.

→ LIKE is used for pattern matching with wildcards.

- % matches any number of characters.

ex:-
-- Names starting with 'A'
Select * from Employees
WHERE Name LIKE 'A%';

-- Names ending with 'n'
Select * from Employees
WHERE name LIKE '%n';

28) What is the IN operator?
→ The IN operator is used to match any value from a list of values.
ex:-
select * from Employees
where department IN ('HR', 'Finance');

29) How do you use the NULL operator?
→ Use IS NULL or IS NOT NULL to check for null values.
ex:-
-- Employees with no departments assigned
select * from Employees
where departments is NULL;

-- Employees with departments assigned
select * from Employees
where department IS NOT NULL;

30) What is the difference between AND and OR operators?

⇒ • AND means both conditions must be true.
• OR means atleast one condition must be true.

ex:-
```
select * from employees
where department = 'HR' AND salary > 3000;

select * from employees
where department = 'HR' OR salary > 3000;
```

31) What are aggregate functions? Give Examples.

⇒ Aggregate functions perform calculations on multiple rows and return a single value.

Common aggregate functions:
- COUNT()
- SUM()
- AVG()
- MAX()
- MIN()

32) What is the COUNT() function?

⇒ COUNT() returns the number of rows that match a condition.

ex:-
```
select COUNT(*) from Employees;
```

```
select COUNT(*) from Employees
where department = 'HR';
```

33.) Explain the SUM() function.
→ SUM() returns the total sum of a numeric column.
ex:-
```
select SUM(Salary) from Employees;
```

34.) What is the AVG() function?
→ AVG() returns the average value of a numeric column.
ex:-
```
select AVG(Salary) from employees;
```

35.) How does the MAX() function work?
→ MAX() returns the highest value in a column.
ex:-
```
select MAX(Salary) from Employees;
```

36.) What is the MIN() function?
→ MIN() returns the lowest value in a column.
ex:-
```
select MIN(Salary) from Employees;
```

37) Explain string functions in MySQL
→ String functions are used to manipulate text (string) values.
Common string functions:-
- CONCAT()
- SUBSTRING()
- LENGTH()
- LOWER() / UPPER()
- TRIM()

ex:-
-- convert name to lowercase
select LOWER(Name) from Employees;

38) What is the CONCAT() function?
→ CONCAT() joins two or more strings together.
ex:-
- combine first and last name
select CONCAT(first_name, ' ', last_name) AS fullname
from Employees;

39) How do you use the SUBSTRING() function?
→ SUBSTRING() extracts parts of a string.
Syntax:-
SUBSTRING(string, start_position, length)

Example:-
select substring(Name, 1, 3) from Employees;
(-- Get first 3 characters of a Name)

40) What is the NOW() function?
→ Now() returns the current date and time.
example:

```
-- show current timestamp
select Now();
```

5) User-Defined Functions (UDFs)

41) What is a user-defined function (UDF) in MySQL?
→ A UDF is a custom function created by the user to perform a specific task. It works like a built-in function but is written by the user.

Use:- When the same calculation or logic is needed in many queries.

42) How do you create a UDF?
→ We can create a UDF using the CREATE FUNCTION statement.
Ex.:

```
CRATE function add_numbers(a INT, b INT)
RETURNS INT
DETERMINISTIC
RETURN a+b;
```

43) What is the syntax for calling a UDF?
→ We can call a UDF like a regular function in a SELECT QUERY.

Ex:- SELECT add_numbers(5, 3); -- Output: 8

44) Can UDFs return multiple values?
→ No, UDFs return only one value.
To return multiple values, you should use stored procedures, not UDFs.

45) What are the advantages of using UDFs?
→ Reusability - Write once, use many times
Clean and readable SQL queries.
Helps avoid repeating the same logic.
Better maintenance of code

~~* Logical Operators *~~

6.) Views

46) What is a view in MySQL?
→ A view is a virtual table based on a SQL SELECT query. It does not store data, but shows data from other tables.

47) How do you create a view?
→ Syntax:

```
CREATE VIEW view_name AS
SELECT Column1, Column2
FROM table_name
WHERE condition;
```

Example :-

```
CREATE VIEW high_salary_employee AS
SELECT name, salary FROM employees
WHERE salary > 50000;
```

48) What is the difference between a view and a table?

Table	View
• Stores actual data	• Does not store data.
• Created using CREATE TABLE	• Created using CREATE VIEW
• Can insert/update/delete directly.	• May or may not allow update
• Takes storage space	• Takes very little storage

49.) Can you update a view? If yes, how?

→ Yes, we can update a view if:
- It's based on a single table
- It doesn't have GROUP BY, DISTINCT, JOIN, etc.

Ex:-

```
UPDATE high_salary_employees
SET Salary = 60000
WHERE name = 'John';
```

50.) How do you drop a view?

→ Syntax:

```
DROP VIEW view_name;
```

Ex:-

```
DROP VIEW high_salary_employees;
```

7.) (Common Table Expressions (CTE)

51.) What is a Common Table Expression (CTE)?

→ A CTE is a temporary result set that you can refer to within a SELECT, INSERT, UPDATE or DELETE query.

52) How do you create a CTE?

→ Syntax:

```
WITH cte_name AS (
    SELECT ...
)
SELECT * FROM cte_name;
```

Ex:-

```
WITH top_students AS (
    SELECT student_name, marks FROM students WHERE
                                    marks > 90
)
SELECT * FROM top_students;
```

53) What is the difference between a CTE and a subquery?

→

CTE	Subquery
• Easier to read and reuse	• Not reusable
• Defined before main query	• Used inside the main query
• Can be recursive	• Not recursive

54) Can you use a CTE recursively?

→ Yes, CTEs can be recursive, which means they can refer to themselves.

Ex:- (printing numbers from 1 to 5):

```
WITH RECURSIVE numbers AS (
    SELECT 1 AS n
    UNION ALL
    SELECT n+1 FROM numbers WHERE n<5
```

)

SELECT * FROM numbers;

55) How do you reference a CTE in a query?
→ Just use its name like a table in the main query.
Ex:-

```
WITH high_salary AS (
    SELECT name, salary FROM employees WHERE
                    salary > 50000
)
SELECT * FROM high_salary;
```

8) Joins :-

56) What is a Join in SQL?
→ A JOIN combines rows from two or more tables based on a related column (usually a foreign key)

57) Explain the different types of joins.
→

Join Type	Description
• INNER JOIN	• Only matching rows from both tables
• LEFT JOIN	• All rows from left table + matching from right
• RIGHT JOIN	• All rows from right table + matching from left.
• Full outer JOIN	• All rows from both tables (not supported in mysql directly)

- CROSS JOIN
  - Every row from one table joins with all rows of the other.
- SELF JOIN
  - Table joined with itself.

58.) What is an INNER JOIN?
→ Shows only rows that have matching values in both tables.
Ex:-
```
SELECT s.name, c.course_name
FROM student s
INNER JOIN courses c ON s.course_id = c.course_id;
```

59.) What is a LEFT JOIN?
→ Shows all rows from the left table, even if there's no match in the right table.

Ex:-
```
SELECT s.name, c.coursename
FROM student s
LEFT JOIN courses c ON s.course_id = c.course_id;
```

60.) What is a RIGHT JOIN?
→ Shows all rows from the right table, even if there's no match in the left table.
Ex:-

```
select s.name, c.course_name
from students s
RIGHT JOIN courses c ON s.course_id = c.course_id;
```

61.) What is a FULL OUTER JOIN?
→ Returns all rows when there is a match in either table.
NOTE: MySQL doesn't supports it directly, but you can simulate it using UNION.

Ex:-

```
select s.name, c.course_name
from students s
left Join courses c ON s.course_id = c.course_id
UNION
select s.name, c.course_name
from students s
Right Join courses c ON s.course_id = c.course_id;
```

SAI PRAKASH | Page No.
Date / /

62) How do you perform a CROSS JOIN?
→ A CROSS JOIN returns all combinations of rows between two tables.
Ex:-
```
SELECT * from shirts
CROSS JOIN colors;
```
This gives all combinations of shirts and colors.

63) What is a self-join?
→ A self-join is when a table is joined with itself.

Ex:-
```
select A.name AS Employee, B.name as Manager
from employees A
Join employees B ON A.manager_id = B.employee_id;
```

64) How do you join multiple tables?
→ Just keep adding Join statements.

Ex:-
```
select o.order_id, c.name, p.product_name
from orders o
join customers c ON o.customer_id = c.customer_id
Join products p ON o.product_id = p.product_id;
```

65.) What is the difference between a join and a subquery?

→

Join	Subquery
• Combines multiple tables	• Query inside another query
• More efficient for large data	• Slower with complex logic
• Good for combining data	• Good for filtering or comparing.

9.) Subqueries

66.) What is a subquery?

→ A subquery is a query inside another query.
It helps to use results of one query in another query.

67.) How do you write a subquery in the SELECT statement?

→ We can put a subquery inside SELECT to calculate values.

Ex:-

```
Select name, (select avg (salary) from employees) AS
                                    avg_salary
from employees;
```

68) Can you use a subquery in the WHERE clause?
→ Yes, its common for filtering data.

Ex:-
```
select name, salary
from employees
where salary > (select AVG(salary) from employees);
```

69) What is corelated subquery?

→ A corelated subquery depends on the outer query's value for each row.
It runs again for every row in the outer query.

Ex:-
```
select e1.name, e1.salary
from employees e1
where e1.salary > (
    select AVG(e2.salary)
    from employees e2
    where e2.department_id = e1.department_id
);
```

70) How do you handle subqueries that return multiple rows?

→ Use operators like:
- IN (matches any value in the list)
- ANY / SOME (matches any value)
- ALL (matches all values)

Ex:- select name
from employees
where department_id IN (select department_id from departments where location = 'mumbai');

10) Stored Procedures

71) What is a stored procedure?

→ A stored procedure is a saved group of SQL statements that can be executed as a program.

72) How do you create a stored procedure in MySQL?

→

Ex:-

```
DELIMITER //
CREATE PROCEDURE get_all_employees()
BEGIN
  SELECT * from employees;
END //
DELIMITER ;
```

73) What is the syntax for calling a stored procedure?

→ Ex:-
CALL get_all_employees();

74) Can stored procedures accept parameters?

→ Yes, we can pass values into them.

Ex:-
```
DELIMITER //
CREATE PROCEDURE get_employees_by_department(IN
                                    dept_id INT)

BEGIN
  select * from employees where department_id=dept_id;
END //
  DELIMITER ;

CALL get_employees_by_department(3);
```

75) What are the advantages of using stored procedures?

→ Reusable - Write once, use many times
Secure - Can restrict direct access to tables
Faster - Runs on the database server
Easy to maintain - Logic stored in one place.

11) Triggers

76) What is a trigger in MySQL?

→ A trigger is a set of SQL statements that automatically runs when a certain event (Insert, update, delete) happens on a table.

77) How do you create a trigger?

→ Ex:-

```
CREATE TRIGGER before_insert_employee
Before INSERT ON employees
FOR EACH ROW
BEGIN
    SET NEW.joining_date = NOW();
END;
```

This trigger sets the joining date automatically before inserting a new employee.

78) What are the different types of triggers?

→ 1. BEFORE INSERT - Runs before inserting data
2. AFTER INSERT - Runs after inserting data
3. BEFORE UPDATE - Runs before updating data
4. AFTER UPDATE - Runs after updating data
5. BEFORE DELETE - Runs before deleting data
6. AFTER DELETE - Runs after deleting data

SAFFRON | Page No.
Date / /

79) Can a trigger call a stored procedure?
→ Yes, but the procedure cannot have transactions (like COMMIT or ROLLBACK) inside a trigger.

Ex:-

```
CREATE TRIGGER after_new_order
AFTER INSERT ON orders
FOR EACH ROW
BEGIN
    CALL update_inventory(NEW.product_id);
END;
```

80) What is the difference between a trigger and a stored procedure?

→

Trigger	Stored Procedure
• Runs automatically	• Runs when called using CALL
• Linked to a table event	• Can be linked to any logic
• Cannot take parameters	• Can take parameters
• Mostly for automation	• For reusable logic.

12) Data Control Language (DCL)

8.1) What is Data Control Language (DCL)?
→ DCL is used to control access to the database.
Main Commands: GRANT, REVOKE.

8.2) What is the purpose of the GRANT command?
→ It gives permissions to a user.
Ex:-
GRANT SELECT, INSERT ON Company.* TO 'john'@'local host';
This allow user john to SELECT and INSERT in all tables of the company database.

8.3) How do you revoke ~~prite~~ privileges using the REVOKE command?
→ Ex:-
REVOKE INSERT ON Company.* FROM 'john'@'local host'
→ This removes the INSERT privilege from john.

8.4) What is the difference between a user and a role in mysql?
→

User	Role
• An individual account	• A collection of privileges
• Permissions assigned directly	• Permissions assigned to a role, the
• Ex:- 'edwin'@'local host'	• Ex:- manager_role with SELECT, UPDATE.

85) How do you create a new user in MySQL?
→ Ex:-
CREATE USER 'edwin'@'localhost' IDENTIFIED BY 'password123';

13) Transaction Control language (TCL)

86) What is Transaction Control language (TCL)?

→ TCL commands are used to manage transactions in a database.
They control how and when changes are saved.
Main TCL commands:
COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION

87) What is the purpose of the COMMIT command?
→ The COMMIT command saves all changes made during the current transaction permanently in the database.
Ex:-
START TRANSACTION;
UPDATE account SET balance = balance - 1000 where account_id = 1;
UPDATE accounts SET balance = balance + 1000 where account_id = 2;
COMMIT; -- Saves both changes.

88.) How do you use the ROLLBACK command?
→ ROLLBACK undoes all changes made in the current transaction (before commit).
Ex:-
```
START TRANSACTION;
DELETE FROM employees WHERE id = 5;
ROLLBACK; -- Cancels the delete
```

89.) What is the SAVEPOINT command?
→ SAVEPOINT creates a checkpoint inside a transaction so you can roll back to that point without undoing the whole transaction.

Ex:-
```
START transaction;
UPDATE accounts SET balance = balance - 500 where
                    accounts_id = 1;
SAVEPOINT step 1;
UPDATE accounts SET balance = balance + 500 where
                    account_id = 2;
ROLLBACK TO step 1; -- Undo only the second update
COMMIT;
```

90.) How do you set the transaction isolation level?
→ The isolation level controls how transactions affects each other's data.
Ex:-
```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION
  -- Your queries here
COMMIT;
```

11) Types of Databases

10) What are the different types of databases?

→ Common types:

1. Relational Database :- Stores data in tables with rows & columns (e.g. MySQL, PostgreSQL)
2. NoSQL Database :- Stores data in formats like documents, key-value pairs, graphs, or columns (e.g., MongoDB)
3. Distributed Database :- Data is stored across multiple locations.
4. Cloud Database :- Hosted on cloud platforms.
5. Graph Database :- Focuses on relationships (e.g., Neo4j).

12) What is the difference between SQL and NoSQL databases?

→

SQL (Relational)	NoSQL (Non-Relational)
• Uses structured tables	• Uses flexible formats
• Follows a fixed schema	• Schema-less or flexible schema
• Best for complex queries	• Best for big data, high speed
• Ex:- MySQL, Oracle	• Ex:- MongoDB, Cassandra

13) What are some examples of NoSQL databases?

→
- MongoDB (Document-based)
- Redis (Key-value)
- Cassandra (Column-based)
- Neo4j (Graph-based)

94.) What is a distributed database?
→ A distributed database stores data on multiple servers or locations but appears as a single database to users.
Ex:- Google's Bigtable, Amazon DynamoDB.

95.) What is a cloud database?
→ A cloud database is hosted on cloud computing platform (like Aws, Azure, Google cloud).

Advantages:- Scalability, accessibility, less maintenance

15.) Database Management System (DBMS)

96.) What is a DBMS?
→ A DBMS is software that help store, manage and retrieve data efficiently.
Ex:- MySQL, PostgreSQL, MongoDB.

97.) What are the functions of a DBMS?
→ Store and retrieve data
Manage users & permissions
Ensure data integrity
Handle transactions
Provide Backup & recovery.

98) What is the difference between a DBMS and an RDBMS?

→

DBMS	RDBMS
• Stores data in files or simple tables.	• Stores data in related tables.
• No relationships between tables.	• Relationships defined using foreign keys
• Ex:- MS Access (basic)	• Ex:- MySQL, PostgreSQL

99) What are some popular DBMS software?

→ MySQL
PostgreSQL
Oracle Database
Microsoft SQL server
MongoDB (NoSQL)

100) What is data integrity, and how does a DBMS ensure it?

→ Data integrity means the data is accurate, consistent and reliable.
A DBMS ensures it using:
• Constraints
• Transactions
• Validation rules