SQL Interview Q&A

① What is SQL? How is it diff from MySQL or PostgreSQL?

→ SQL is a language used to communicate with database – to insert, retrieve, update or delete data.

~~My SQL~~ ~~Postgree SQL~~

~~②~~ • My SQL and PostgreSQL are database management System (DBMS) that use SQL
• SQL is like English grammer, while MySQL / Postgre SQL are book's written using that grammer

② What are the diff types of SQL statement?
→ ① DDL (Data Definition Language) – Create, Alter, Drop.
② DML (Data Manipulation Language) – Insert, Update Delete
③ DQL (Data Query Language) – Select
④ DCL (Data Control Language) – Grant; Revoke
⑤ TCL (Transaction Control Language) – Commit Roll Back, Save point.

③ Explain the diff b/w Where & Having?
→ Where is used before Groupby, to filter row's.
Having is used after ~~Grouping~~ ~~by~~ Groupby, to filter group's

④ What are Primary key, Foreign key, Unique and Check Constraint's?
→ ① Primary key → Uniquely identifies each row's, only one pertable.
② Foreign key → Link's one table to another.
③ Unique → Ensure's all values in a Column are different.
④ Check → Validate data before inserting (e.g age > 18);

5) What is the diff b/w Delete, Truncate and Drop?

→ Delete → • Can Delete Data
- Can't roll back
- ~~Don't~~ Doesn't remove table structure

Truncate → • Can Delete Data
- Can be roll back
- Doesn't remove table structure

Drop → • Can Delete Data
- Can be roll back
- It removes table structure.

6) What is normalization? Explain diff normal form's?

→ Normalization is a process of organizing data to reduce redundancy & improve data integrity.
- 1NF → Remove repeating group's
- 2NF → Remove partial dependency (Based on primary key)
- 3NF → Remove transitive dependency (non key depending on non key).

7) What is denormalization and when it's useful?

→ Denormalization is ~~process~~ the opposite of normalization - it add's ~~redundan~~ redund-ancy to improve read performance.
Used ~~to~~ when :-
- Fast reading is more important than saving space.
- In reporting or data warehousing system's.

⑧ Explain the diff b/w char and Varchar?

→ Char → • Fixed length
• More storage Space
• Slightly faster.

Varchar → • Variable length
• Less Storage Space
• Slightly slower

Ex :- Char(20) always uses 20 bytes;
Varchar(20) uses only needed space.

⑨ What are ACID ~~[illegible]~~ Properties in databases?

→ ① Atomicity :- All or nothing (transaction fully completes or roll's back).

② Consistency :- Data remain's valid before and after transaction.

③ Isolation :- Transaction don't affect each other

④ Durability :- Once Committed, data stays safe.

⑩ What is diff b/w Inner, Left, Right & Full Join?

→ ① Inner Join :- only matching row's from both tables

② Left Join :- All row's from left + matching from right.

③ Right Join :- All row's from right + matching from left.

④ Full Join :- All row's from both table (match or not).

(11) Find the 2nd higlest salary from Employee table?
→ Select Max (Salary)
From Employee
where salary < (Select Max (Salary) from Employee);

(12) Department-wise avareage salary?
→ Select department, AVG (salary) As avg_salary
From Employee Group by department;

(13) Retrieve duplicate record's from a table?
→ Select name, count (*) From Employee
Group by name Having Count (*) > 1;

(14) Update a column with 10% tax added?
→ Update Product's set price = price +
(price * 0.10);

(15) Delete only duplicate rows from a table?
→ Delete From Employee where id NOT IN
(Select MIN (id) From Employee Group
By name, Salary);

(16) Customer's who placed more than 5 order's?
→ Select customer_id From order's Group by
Customer_id Having Count (order_id)
> 5;

(17) Join three or more table's?
→ Select orders.order_id, Customer.name,
Products.product_name from order's
Join Customers ON Orders.Customer_id
= Customers. Customer_id
Join Products ON Order.product_id =
Product's.product_id;

(18) What is a subquery? How is it different from Join?

→ A subquery is a query inside another query, often used in where, In, or Select.

→ A Join combines data from multiple tables using common column's.

(19) What is a correlated subquery? Example?

→ A correlated subquery depends on the outer query for each row.

Ex:-

Select e1.name, e1.salary From Employee e1 where where salary > (Select AVG(salary) From Employee e2 where e1.department = e2.department);

(20) Filter data based on a date range?

→ Select * From order's where order_date Between '2024-01-01' And '2024-12-31';

(21) What are WINDOW FUNCTION? Name a few?

→ Window function perform calculation across a set of rows related to the current row, without collapsing rows.

Ex:-

ROW_NUMBER(), RANK(), DENSE_RANK, SUM(), AVG(), COUNT()

(22) Use of RANK(), DENSE_RANK() and ROW_NUMBER?

→ These are raking functions used with Over (Partition by

Select name, salary,
ROW_NUMBER() over (Order BY Salary DESC) AS row_num,
RANK() over (order by Salary DESC) AS rank,
DENSE_RANK() over (Order by Salary DESC) AS dense_rank
From Employee;

(23)

(23) What is CTE (Common Table Expression)?

→ • CTE is a temporary result set you can reference in a SELECT, INSERT, UPDATE, or Delete.
• Difference from subquery
• CTE is reusable, readable and can be recursi-ve.

Ex:- With High Earner's AS (Select * from Employee where Salary > 45000) Select * from High Earner's;

(24) What are stored procedur's? When to use them?

→ A stored procedure is a saved SQL block that can take input's and return result's.
Use when :-
• You need to reuse Complex SQL Logic.
• Improve performance
• Maintain business rules.

Ex:- CREATE PROCEDURE Gethigh Salary()
BEGIN
 Select * From Employee WHERE salary > 5000;
END;

(25) What is a Trigger? Example?

→ A trigger is SQL code that runs automati-cally when an event (INSERT/UPDATE/DELETE) occur's.

Ex:- CREATE TRIGGER update_log
AFTER UPDATE ON Employee
FOR EACH ROW
INSERT INTO log_Table (message)
Values ('Employee record updated');

(26) What is a View? Pros and Cons?

→ A view is a virtual table based on query

Pros :-
- Simplifies complex queries.
- Add's security (hides sensitive column)
- Helps modularize logic.

Cons :-
- Slower than tables
- Cannot always update
- Doesn't store data (unless materialized).

(27) What are indexes? How do they improve Performance?

→ Indexes help speed up searching, sorting and filtering.

Ex :-

```
CREATE INDEX idx_name ON Employee
(name);
```

Improves Performance by reducing the amount of data scanned.

(28) What is a materialized view?

→ Materialized view stores the result of a query physically (unlike normal view).
- Faster for reporting.
- Needs refresh to stay updated.

Ex :-

```
CREATE Materialized VIEW dept_salary
AS Select department, AVG(salary)
 From Employee Group by department
```

(29) What are transaction's? COMMIT, ROLLBACK, SAVE POINT.

- Transaction :- Group of operation's executed as a single unit.
- COMMIT :- Save all changes.
- ROLL BACK :- Undo changes.

Save Point :- Set a point to roll back to

Ex:-
```
START TRACTION;
UPDATE Employee SET Salary = Salary +
5000;
Save point before_bonus;
UPDATE Employee SET Salary = Salary +
10000;
ROLL BACK To before_bonus;
COMMIT;
```

(30) Aggregate Function's (with Example)?

→ • SUM () :- Total, Select SUM (salary) From Employee;
• AVG () :- Average, Select AVG (Salary) From Employee;
• MAX () :- Maximum, Select MAX (Salary) from Employee;
• MIN () :- Minimum, Select MIN (Salary) From Employee;
• COUN () :- Count row's, Select Count (*) From Employee;

(31) How can you optimize a slow-running SQL ~~Q~~ Query?

→ • Use indexes on Column used in WHERE, Join, order by
• Avoid Select *, Select only required Column
• Use Exists instead of IN.
• Avoid subqueries in SELECT, use Join instead
• Analyze with EXPLAIN.

(32) What is the Explain or Explain Plan statement used for?

→ It show's how the SQL engine executes your query:-
- Table scan vs index usage.
- Join types
- Estimated cost
- Helps identify slow parts of the query.

(33) How does indexing affect INSERT, UPDATE, DELETE performance?
→ • Speed up Select's
- Slow's down INSERT/UPDATE/DELETE because the index must also be updated.

(34) What is a Composite index and when to use it?
→ A Composite index is an index on multiple Column
- Use when queries filter or sort on both Column

Ex:-

```
CREATE. INDEX idx_dept_salary ON Employee
 (department, salary);
```

(35) What is normalization ~~B~~ overhead and how to deal with it?
→ • Overhead :- Too many small tables → many Joins → slower queries
- Solution :- Use denormalization where performance is more important than storage.

(36) How do you avoid Cartesian product's in Join?
→ • Always use Join Conditions (eg., ON or where)
- Don't forget to match keys in joins.

Ex:-

```
Select * From A Join B on A.id = B.id;
```

(37) What is partitioning in SQL?
→ Partitioning splits a table into smaller pieces for performance and mangeability.

Types:-

ⓐ Range Partitioning.
ⓑ List Partitioning
ⓒ Hash Partitioning.

chandra's

(38) What causes a deadlock in SQL, and how Can you prevent it?

→ • Deadlock :- Two or more queries wait for each other's lock's → stuck

• Prevention :-
ⓐ Access table in the same order
ⓑ keep transactions short
ⓒ Use low isolation levels if safe.

(39) Difference b/w clustered and non-clustered indexes?

Features	Clustered index	Non-clustered index
ⓐ Data Storage	Sort's and stores actual row's	Stores pointer's to row's
ⓑ Count per table	Only one	Can have many
ⓒ Speed	Faster for range queries.	Better for lookup.

(40) What tools do you use to monitor SQL query Performance?

→ • MySQL :- Explain, Show Profile, slow ~~[illegible]~~ Query Log
• PostgreSQL :- Explain Analyze, Pg_stat_statements.
• SQL Server :- Execution Plans, SQL Profiler.

(41) Student - Course Grading System - Tables & Relationship's?

→ Tables :-
• Students (student_id, name, age, ...)

- Courses (Course_id, Course_name, ...)
- Enrollment's (enrollment_id, student_id, course id, grade)

Relationships :-

- Many-to-Many between Student's and Courses through Enrollments.

42 Storing & Retrieving Attendance (Scalable way)?

→ Tables :-

- Employees (employee_id, name, ...)
- Attendance (attendance_id, employee_id, date, Check_in, check_out)

Tips :-

- Use indexes on employee_id and date.
- Partition table by date (monthly / yearly) for scalability.

43 Tracking Overdue Books & Fines?

→ Tables :-

- Book's (book_id, title)
- Borrowings (borrowing_id, user_id, book_id, borrow_date, due_date, return_date)

SQL Example :-

```
Select used_id, book_id, DATEDIFF (CURDATE
(), due_date) As ~~[illegible]~~ overdue_days
From ~~[illegible]~~ Borrowings
Where return_date IS NULL AND due_date
 < CURDATE ();
```

44 Missing Records After Failed Update?

→ Steps :-

(a) Check logs / transaction history.
(b) Roll back if using transaction
(c) Restore from backup if necessary.
(d) Identify failed query and rerun carefully with checks.

45 Role Based Access in SQL?

→ • Use GRANT/REVOKE statements
Ex :-
GRANT Select ON student_data To role_student;
REVOKE UPDATE ON student_data From role_student;
• Create user roles based on permissions.

(46) Clean Raw CSV with SQL?

→ • Load with LOAD DATA INFILE.
• Clean using :-
  • TRIM () for spaces.
  • REPLACE () for invalid character's.
  • Remove duplicate with Distinct or ROW_NUMBER () + DELETE
• Staging table approach is best.

(47) Monthly Retention Calculation?

→ • Retention = user's who logged in again after signup month.
Ex :-
Select signup_month, COUNT (DISTINCT user_id) As retained_users From logins
Where login_date > Date_ADD (signup_date, INERVAL 30 DAY)
Group by signup_month;

(48) Database Security Measures?

→ • Use encryption
• Role based access control
• Regular patches & updates.
• Audit logs & Monitoring.
• Parameterized queries

(49) Daily Backup & Restore Plan's?

→ • Backup :-

- mysqldump for logical backups.
- cron job for automate daily dumps.

• Restore :-

```
mysql -u user -p dbname < backup.sql
```

(50) Best Practices for Production SQL?

→ • Use indexed column in where.
- Avoid Select * , use specific Column.
- Test queries before running.
- Use transactions for bulk updates/ deletes.
- Limit joins / sub squeries for performance.

[John Abhishek. A]