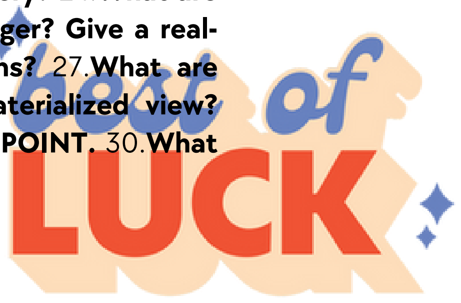
! TOP 50 INTERVIEW QUESTIONS FOR SQL !

1.**What is SQL? Howis it different from MySQL or PostgreSQL?**
2.**What are the different types of SQL statements?**
3.**Explain the difference between WHERE and HAVING.**
4.**What are PRIMARY KEY, FOREIGN KEY, UNIQUE, and CHECK constraints?**
5.**What is the difference between DELETE, TRUNCATE, and DROP?**
6.**What is normalization? Explain different normal forms.**
7.**What is denormalization and when is it useful?**
8.**Explain the difference between CHAR and VARCHAR.**
9.**What are ACID properties in databases?**
10.**What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN?**
11.**Write a query to find the second highest salary from an Employee table.**
12.**Write a query to get the department-wise average salary.**
13.**How would you retrieve duplicate records from a table?**
14.**How do you update a column with a calculation (e.g., 10% tax added)?**
15.**How would you delete only duplicate rows from a table?**
16.**Write a query to list customers who have placed more than 5 orders.**
17.**Write a query to join three or more tables.**
18.**What is a subquery? How is it different from a JOIN?**
19.**What is a correlated subquery? Give an example.**
20.**How do you filter data based on a date range?**
21.**What are WINDOW FUNCTIONS? Name a few.**

22.**What is the use of RANK(), DENSE_RANK(), and ROW_NUMBER()?** 23.**What is a Common Table Expression (CTE)? How is it different from a subquery?** 24.**What are stored procedures? When should they be used?** 25.**What is a trigger? Give a real-world example.** 26.**What is a VIEW? What are its pros and cons?** 27.**What are indexes? How do they improve performance?** 28.**What is a materialized view?** 29.**What are transactions? Explain COMMIT, ROLLBACK, and SAVEPOINT.** 30.**What are aggregate functions? List a few with examples.**



31.**How can you optimize a slow-running SQL query?**
32.**What is the EXPLAIN or EXPLAIN PLAN statement used for?**
33.**How does indexing affect INSERT, UPDATE, and DELETE performance?**
34.**What is a composite index and when should it be used?**
35.**What is normalization overhead and how do you deal with it?**
36.**How do you avoid Cartesian products in JOINs?**
37.**What is partitioning in SQL?**
38.**What causes a deadlock in SQL, and how can you prevent it?**
39.**What is the difference between clustered and non-clustered indexes?**
40.**What tools do you use to monitor SQL query performance?**
41.**You are asked to design a student-course grading system. What tables and relationships would you create?**
42.**How would you store and retrieve attendance for employees in a scalable way?**
43.**In a library system, how would you track overdue books and fines using SQL?**
44.**What would you do if a production database is missing some records due to a failed update?**
45.**How would you implement role-based access to sensitive information in SQL?**
46.**You are given a raw CSV with dirty data. How would you load and clean it using SQL?**
47.**How would you calculate monthly retention from a user login dataset?**
48.**What measures would you take to secure a database with sensitive data?**
49.**How do you create daily backup and restore plans for a SQL database?**

TOP 50 INTERVIEW QUESTIONS FOR SQL

Qns1. What is SQL? How is it different from MySQL or PostgreSQL

Ans. MySQL/PostgreSQL/SQL Server all these are DBMS software which is used to interact with Database. Database is like Hardware and DBMS is like software and to interact with hardware we need software and DBMS is that software which is used to manage the database and DBMS software are also knows as Database server software. SQL Server, MySQL, PostgreSQL, Oracle these are Database Server Software. To connect with DBMS software, we need a software that is client software. SSMS, MySQLWorkBench, pgAdmin, SQLight are Database Client Software tool. To Connect Database Client Software Tool to Database Server Software, we need a language and that language is SQL. So, SQL stands for Structure Query Language, which is used for communication from Client Tool [SSMS, MySQLWorkBench, pgAdmin,SQLight] to Server Tool [SQL Server, MySQL, PostgreSQL, Oracle].

User ----> SSMS (Client Tool) <----> SQL <----> SQL Server (DBMS- Server Tool) <----> Database
User ----> MySQLWorkBench(Client Tool) <----> SQL <----> MySQL (DBMS- Server Tool) <----> Database
User ----> pgAdmin (Client Tool) <----> SQL <----> PostgreSQL (DBMS- Server Tool) <----> Database

Qns2. What are the different types of SQL statements?

Ans. Based on operations over database, SQL is categorized into following sub-languages

1) DDL (Data Definition Language)
2) DML (Data Manipulation Language)
3) DQL (Data Query Language)
4) TCL (Transaction Control Language)
5) DCL (Data Control Language)

All these five sub-languages contains total 14 SQL Commands/statements

DDL	DML	DQL	TCL	DCL
CREATE	INSERT	SELECT	COMMIT	GRANT
ALTER	UPDATE		ROLLBACK	REVOKE
DROP	DELETE		SAVE TRANSACTION	
TRUNCATE	MERGE			

Qns3. Explain the difference between WHERE and HAVING.

Ans.

WHERE	HAVING
1) Selects specific rows	1) Selects specific groups
2) Conditions applied before GROUP BY	2) Conditions applied after GROUP BY
3) Use WHERE clause, if condition doesn't contains contain aggregate function.	3) Use HAVING clause if condition contains aggregate functions like MIN(), MAX(), SUM() , AVG(), COUNT()

Qns4. What are PRIMARY KEY, FOREIGN KEY, UNIQUE, and CHECK constraints?

Ans. PRIMARY Key, FOREIGN Key, UNIQUE, and CHECK all these are Integrity Constraints. Integrity constraints are set of rules which is used to maintain data integrity i.e. data qaulity or data consistency. This is main used to prevent user to enter invalid/wrong data into table. All these constraints can be applied at time of table schema creation or after the table schema creation also.

1) **UNIQUE Constraint** => It doesn't accept duplicate data and allow only one null.

Syntax:- *<ColumnName> <DataType> UNIQUE*

2) **PRIMARY KEY Constraint**

- Primary key constraint doesn't accept duplicates and nulls.
- It is the combination of unique & not null constraints.
- In tables one column must be there uniquely identify and into that column duplicates and nulls are not allowed, so declare that column with primary key

Syntax:- *<ColumnName> <DataType> PRIMARY KEY*

3) **CHECK Constraint** => Use CHECK constraint on Column when rule based on some condition

Syntax:- *<ColumnName> <DataType> CHECK(condition)*

Ex:- Every new employee sal must be min 3000.

SAL MONEY CHECK(SAL>=3000)

Qns5. What is the difference between DELETE, TRUNCATE, and DROP?

Ans. Drop :- By using Drop command, we can drop/remove Database Object(Table, Database, Triggers, Views etc.) or structure along with data.

Syntax:- *DROP <Databse_Object> <Databse_Object Name>*

Ex:- Drop Database College_DB

Drop Table Employee

Drop Procedure SP_GetEmploye

DELETE	TRUNCATE
1) It's a DML command	1) It's a DDL command
2) Can delete all rows, and specific rows	2) It can delete only all rows, but can't delete specific rows
3) WHERE condition can be used with Delete	3) WHERE condition cannot be used with Truncate
4) Deletes row-by-row	4) Deletes all rows at a time
5) It is slower	5) It is faster
6) It will not release memory immediately	6) It will released memory immediately
7) It will not reset IDENTITY column	7) It will reset IDENTITY column
Syntax: Delete From <TableName> [Where Cond]	Syntax: Truncate Table <TableName>

Qns6. What is normalization? Explain different normal forms.

Ans. Normalization is the process of decomposing tables with redundency into number of well structured tables.

Normalization is set of rules and each rule is called one normal form and there are six normal forms.

1NF (1st Normal Form)
2NF (2nd Normal Form)
3NF (3rd Normal Form)
BCNF (Boyce-Codd Normal Form)
4NF (4th Normal Form)
5NF (5th Normal Form)
6NF (6th Normal Form)

1NF :- A table said to be in 1NF if there are no multivalued attributes in it or all the attributes in table are atomic (single).

2NF :- A table said to be in 2NF

1) If it is in 1NF.
2) If there are no partial dependencies in it.

Full dependency => In table if non key fields depends on key field then it is called full dependency.

```
Ex:-    R(A,B,C,D)          A  =>  pk
        -  --- -------
        |        |
     Table     Field
     Name      Name

   A  ====>  B,C,D    (full dependency)
```

Real life Example => Bank Table, Balance is Depends upon Accno. Here Accno is primary key.

Partial dependency => If table consist composite primary key and if non key field depends on part of the key field then it is called partial dependency

```
Ex:-    R(A,B,C,D)        A,B   =>  pk
        A,B =======> C    (full dependency)
        B ========> D     (partial dependency)
```

3NF :- A table is said to be in 3NF, if

1) If it is in 2NF.
2) If there are no transitive dependencies exists in the table.
3) if there are any derived attributes we can permanently remove them.

Transitive dependency => If non key field depends on another non key field then it is called transitive dependency.

```
Ex:-  R(A,B,C,D)            A   => pk
        A ======> B,C  (full dependency)
        C ======> D    (transitive dependency)
```

BCNF :- A table is said to be in BCNF, if
1) If it is in 3NF.
2) for any dependency A → B, A should be a super key.

Super key => A column (or) combination of columns which are uniquely identifying a row in a table is called as super key.
Ex:- stid, stid+mailid, mailid+reg_number, stid + mailid + reg_number

Candidate key => A minimal super key which is uniquely identifying a row in a table is called as candidate key.
Ex:- stid, mail, reg_number

Note => A super key which is subset of another super key, then the combination of super keys are not a candidate key. In db design only db designer uses super key and candidate key. That mean first designers select super keys and then only they are selecting candidate keys from those super keys.

4NF :- A table is said to be in 4NF, if
1) If it is BCNF.
2) A table does not contain more than one independent Multi-valued attribute / Multi Valued Dependency.

Multi-valued Dependency => In a table one column same value match with multiple values of another column is called as multi valued dependency.

5NF :- If a table having multi valued attributes and also that table cannot decomposed into multiple tables is called as fifth normal form. Generally in 4NF resource table some attributes are not logically related where as in 5NF resource table all attributes are related to one to another.

Qns7. What is denormalization and when is it useful?

Ans. When your required data is in multiple tables, and to retrive the data you join the tables using JOINs concept in SQL. This joining process is called Denormalization.

Qns8. Explain the difference between CHAR and VARCHAR

Ans.

CHAR	VARCHAR
1) Recommended to use for fixed length character column. Ex:- Gender, Country code, state code etc.	1) Recommended to use for variable length columns. Ex:- Name, Address, Email Id, State Name, Country etc.
2) Memory wastage	2) No Memory Wastage

Qns9. What are ACID properties in databases?

Ans. 1) All databases are having ACID properties by default.

2) By using these ACID properties databases are maintain and manage the transactions with accurate and consistency manner.

A - Atomicity
C - Consistency
I - Isolation
D - Durability

Atomicity

- Atomic means "single".
- To make all operations as a single transaction which is commit / rollback by database server

Ex:- Withdraw Transaction:

[ATM]

Step 1: Insert ATM card
Step 2: Select language
Step 3: Click on banking
Step 4: Click on withdraw option
Step 5: Enter amount : 10000
Step 6: Select current / save account
Step 7: Enter pin no : xxxx
Step 8: Yes/No

Consistency

- To maintain accurate information before / after a transaction.

Ex:

x-customer	**y-customer**
Main Bal : 10000 (before transaction)	Main Bal : 4000 (before transaction)
Transfer : 2000 (debit)	: 2000 (credit)
Bal amt : 8000 (after transaction)	Bal amt : 6000 (after transaction)
: 2000(credit)	: 2000(debit)
10000	4000

Isolation

- It will treat every transaction is independent transaction.

Durability

- Once a transaction is committed then we cannot rollback at any level.

Qns10. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN?

Ans. **INNER JOIN** :- Inner join will join the tables and only returns matching records based on join condition

LEFT JOIN :- Left join will join tables based on common column and returns both matching and unmatching records From Left table and Matching records only from Right Table based on join condition.

RIGHT JOIN :- Right join will join tables based on common column and returns both matching and unmatching records From Right table and Matching records only from Left Table based on join condition.

FULL JOIN :- Full join will join tables based on common column and returns both matching and unmatching records From both side Tables based on join condition.

Qns11. Write a query to find the second highest salary from an Employee table.

Ans. Select Max(Sal) From Employee Where Sal != (Select Max(Sal) From Employee)

Qns12. Write a query to get the department-wise average salary.

Ans. Select DEPTNO, Avg(Sal) as AvgSal From Employee Group By DEPTNO

Qns13. How would you retrieve duplicate records from a table?

Ans. To retrieve duplicate records from a table, we can use "GROUP BY" clause combined with HAVING COUNT(*) > 1. This identifies groups of rows that share the same values in one or more columns and appear more than once.

Ex:- SELECT name, COUNT(*) AS count FROM students GROUP BY name HAVING COUNT(*) > 1

Qns14. How do you update a column with a calculation (e.g., 10% tax added)?

Ans.
```
Update Employees SET SAL= CASE
                            When ((ISNULL(SAL,0)-ISNULL(SAL,0))*0.1)<0 Then 0
                            Else (ISNULL(SAL,0)-ISNULL(SAL,0)*0.1)
                            End
```

Qns15. How would you delete only duplicate rows from a table?

Ans. We can delete the duplicate rows with the help of using CTE, Row_Number() Function, Partition By, and Order By Clause.

Ex:- With E as (Select Row_Number() Over(Partition By Eno,Ename,Dno Order By Eno) as Rno,Eno,Ename,Dno From Emp44) Delete From E Where Rno>1

Qns16. Write a query to list customers who have placed more than 5 orders.

Ans. Select O.Ordid,C.Cname, Count(O.Prodid) As Orders_Placed From Cust as C join
Sales as S ON C.Custid=S.Custid join Order_Details as O ON S.Prodid=O.Prodid
Group By O.Ordid,C.Cname Having Count(*)>5

Qns17. Write a query to join three or more tables.

Ans. Select P.Programme_ID,SM.SemesterName,S.Subject_ID,S.Subject_Name,S.Credits From Programme as P Join Subject as S ON P.Programme_ID=S.Programme_ID join Semesters as sm ON S.SemesterID=SM.SemesterID Order By P.Programme_ID,SM.SemesterName

Qns18. What is a subquery? How is it different from a JOIN?

Ans. A query inside another query is called Subquery or Nested Query. One query is called Inner/Child/Sub Query and other query is called Outer/Parent/Main Query. We use subquery, when condition value in WHERE clause is based on unknown value.

Ex:- **Give the list of employees, who earning salary more than '<u>Blake</u>' employee.**
|=> Salary is Unknown

Then here first we need to find out the salary of blake then we can able to employee list who earning more than Blake.

Ex:- <u>Select * From Employee Where Sal ></u> (Select Sal From Employee Where Ename='Blake')
|
Outer Query
|
Inner Query [Return Salary of Blake]

<u>Difference between Subquery & JOIN</u>

- To display data from one table and condition based on another table then we can use join or sub-query.
- To display data from two tables then compulsory use join and this is not possible with subj- query.

Qns19. What is a correlated subquery? Give an example.

Ans. Correlated Subquery :- If inner query refers the value of outer query then it is called correlated subquery.

Ex:- **Employees earning more than avg sal of their dept ?**

Select * From Employee as **X** Where Sal > (Select Avg(Sal) From Employee Where Deptno = **X.Deptno**)
|
Outer Query Reference
|
Inner Query refers to Outer Query Deptno col

- In Correlated subquery, excution starts from Outer Query & Inner Query is executed number of times depends on number of rows return by outer query.

- In Correlated subquery, both inner & outer query taking values as input to each other, that's why this is called Correlated subquery.

Qns20. How do you filter data based on a date range?

Ans. To filter data based on range we used Combination of Relational (>=, <=) & Logical operator (AND) OR Combination of Logical operator (AND) and Between Operator.

Ex:- **Display Employees earning between 5000 and 10000 ?**

Select * From Employees Where Sal >= 5000 AND Sal <= 10000

OR

Select * From Employees Where Sal Between 5000 AND 10000

Qns21. What are WINDOW FUNCTIONS? Name a few.

Ans. Window functions in SQL helps you to perform calculations across a set of rows that are related to the current row without collapsing the result set. window functions preserve individual rows while adding calculated columns. The term "window" refers to a frame of rows defined by the Over( ) clause.

Some Window Functions are:

1) **Row_Number( )**
2) **Rank( )**
3) **Dense_Rank( )**
4) **Lag( )**
5) **Lead( )**

Qns22. What is the use of RANK(), DENSE_RANK(), and ROW_NUMBER()?

Ans. Rank( ) and Dense_Rank( )

- Both functions used to find ranks
- Ranking is based on some column
- For rank functions data must be sorted either Ascending or Descending

Rank( ) => Rank function generates gaps but dense_rank will not generate gaps means, ranks may not be in sequence if two or more person got same rank. It is generally used for international ranking.

Syntax:- *Rank( ) Over(Order By ColumnName Asc/Desc)*

Ex:- **Find the ranks of the employees based on sal and highest paid should get 1st rank ?**

Select Eno, Ename, Sal, Rank() Over(Order By Sal Desc) as Rnk From Employee

Dense_Rank( ) => Dense_Rank function maintains sequence if two or more person got same rank. It used generally for local ranking like rank based on highest salary, rank based on experience or employee etc.

Syntax:- *Dense_Rank( ) Over(Order By ColumnName Asc/Desc)*

Ex:- **Find the ranks of the employees based on sal and highest paid should get 1st rank ?**

Select Eno, Ename, Sal, Dense_Rank() Over(Order By Sal Desc) as Rnk From Employee

Row_Number( )

- Returns row number i.e. record number/record serial number
- ROW_NUMBER is also based on one or more columns
- For ROW_NUMBER data must be sorted

Syntax:- *Row_Number( ) Over(Order By ColumnName Asc/Desc)*

Ex:- **I want to give row number to every records in a Employees table**.

```
Select Row_Number() Over(Order By Eno Asc) as Rno, Eno, Ename, Sal From Employee
```

Qns23. What is a Common Table Expression (CTE)? How is it different from a subquery?

Ans. - In Derived tables "outer query cannot be dml" and it must be always select. To Solve this, CTE concept introduced.

- Using CTE, we can give name to the query output and we can use that name in another query like select/insert/update/delete.
- CTEs can be use to solve complex queries

Syntax:- *WITH <CTE-Name1> As (SELECT Stmts), <CTE-Name2> As (SELECT Stmts)*
SELECT/INSERT/UPDATE/DELETE Stmts

Ex:- **Delete first 3 rows of Employee Table?**

```
WITH E As (Select Row_Number( ) Over(Order By Eno Asc) as Rno, Eno, Ename,Sal From
Employee) Delete From Employee Where Rno<=3
```

Qns24. What are stored procedures? When should they be used?

Ans. - A stored procedure is a named T-SQL block that accepts some input, performs some action on db, and may or may not returns a value.

- Stored Procedures are created to perform one or more actions like insert/update/delete.
- If there is one time action then procedure is not required. If there is repeated action then procedure is required. In realtime maximum action is repeated action so here we required procedures.
- In realtime, user are not able to access database and tables directly. They are able to access the through Stored procedures & Functions.

Syntax:-
```
CREATE OR ALTER PROCEDURE/PROC <name>
Parameters if any
AS
Begin
        statements
End
```

Ex:-
```
Create OR Alter Procedure SP_GetCourseSubjectDetails
AS
Begin
    Select P.Programme_ID,SM.SemesterName,S.Subject_ID,S.Subject_Name,S.Credits From
Programme as P join Subject as S ON P.Programme_ID=S.Programme_ID join Semesters as sm ON
```

S.SemesterID=SM.SemesterID Order By P.Programme_ID,SM.SemesterName
    End

- Stored Procedure can be Execute from two ways
    1) **SSMS** (Client Tool Sofware)
    2) **Front-End Applications**

- Executing from SSMS (Client Tool Software)
    Syntax:- *EXECUTE <Procedure-name> [Parameter1, Parameter2,...]*
    Ex:- Execute/Exec SP_GetCourseSubjectDetails

Qns25. What is a trigger? Give a real world example.

Ans. - A Trigger is also a named T-SQL Block like procedure but executed implicitly by sql server whenever user submits DML commands.

- Triggers are created for
1) To control DML's

Real life examples :-
-In company, you are allowed to perform Insert/Update/Delete only between 9 to 5 on Monday to Friday
- Allowed to perform Insert/Update/Delete only between 9 to 2 on Saturday
- Not allowed to perform Insert/Update/Delete on Sunday
- Allowed to updates other fields but not EmpId field, because it is primary key field.
- You are allowed to increment the salary but not allowed to decrement the salary.
- In Adhaar card, you can update DOB and Name only two times, then use Triggers.

2) To enforce(Implementing) complex rules and validations

Note:- In SQL Server, rules can be implemented using two ways
    i) **Integrity Constraints**
    ii) **Triggers**

3) To audit day-to-day operations on tables

Real life examples :-
- I want to know who is inserting the data, what time inserting the data, what data he/she is inserting, what data he is updating and who is updating, what are the old value, who is deleting, what time data is deleted, that is called Auditing on table.

- Triggers are two types
    1) **After** Trigger :- executes after executing dml
    2) **INSTEAD OF** Trigger :- executes the trigger instead of executing dml

    Syntax:- *Create OR Alter Trigger <Trigger-Name>*
    ON <Table_Name>
    AFTER / INSTEAD OF <Dml Commands>
    AS
    Stmts---

- Trigger doesn't contain Parameters because, Trigger is not a sub-program. Procedure and Functions are Sub-Programs. Sub-programs can be called in other programs but trigger can not be called. Due to this reason Trigger doesn't contains any parameters.

Ex:- **Create a Trigger to not to allow DML's on "Employees" table on sunday ?**

```
Create OR Alter Trigger Trg_OPNotAllowONSunday
ON Employee
After INSERT,UPDATE,DELETE
AS
        IF(DATENAME(DW,GETDATE())='SUNDAY')
        BEGIN
        Rollback
        Raiserror('Operation is not allowed on Sunday !',16,1)
END
```

Qns26. What is a VIEW? What are its pros and cons?

Ans. - A view is a subset of a table i.e. part of the table.

- A view is a virtual table because it doesn't store data and doesn't occupy memory and it always derives data from base table.
- A view is a representation of a query.
- Views provided security at rows & cols level.
- In real time, permission are not Granted on table to developer. Company will create views, then on that view they grant the permission to User. So User can access specific column and rows.

- View are two types

1) **Simple View** :- A view said to be simple view, if it is created on single table.

Syntax:- *Create View <View-Name> AS Select Stmts*

Ex:- Create View V1 AS Select Empno,Ename,Job,Deptno From Where Deptno=10

2) **Complex View** :- A view said to be complex view

i) If based on multiple tables

ii) If query contains DISTINCT clause, GROUP BY clause, AGGREGATE functions, SET operators, Sub-Quaries.

Syntax:- *Create View <View-Name> AS Select Stmts*

EX:- CREATE VIEW CV1 AS SELECT E.Empno, E.Ename, E.Job, D.Deptno, D.Dname, D.Loc FROM Employees as E INNER JOIN Dept as D ON E.Deptno=D.Deptno

<u>Advantages</u>

- It provide rows and columns level security
- Reduce complexity for Front-developer,middleware developer, which is not expert in writing complex query.

<u>Disadvantage</u>

- We can't perform DML(Insert/Update/Delete) Operations on Complex View.
- It again and again complies and execute, which reduces performance of SQL Server

Qns27. What are indexes? How do they improve performance?

Ans. - Index is also a database object created to improve query performance.
- Index makes data accessing fater.
- Index in db is similar to index in textbook , In textbook using index a particular topic can be located Fastly, In db using index a particular record can be located fastly.
- Indexes created on columns and that column is called index key.

- Indexes created on columns
    1) **Frequently used in where clause**
        Ex:- Empno, Sal, Accno, Phone

    2) **Frequently used in join operation**
        Ex:- I have joined Employees and Dept table with join condition "Deptno" frequently. Better to create index on Deptno

- Types of Indexes
1) **Non clustered Indexes** :- A non clustered index stores pointers(Address) to actual records and table are seperate objects. There are mainly 3 types of Non-Clustered Indexes.

i) **Simple index** :- If index created on single column then it is called simple index.
Syntax:- *Create Index <Index-Name> ON <Table-Name> (<Column-Name>)*
Ex:- Create Index NCI_Sal ON Employee(SAL)

ii) **Composite index** :- IF index created on Muliple columns then it is called Composite index
Syntax:- *Create Index <Index-Name> ON <Table-Name> (<Column-Name1>, <Column-Name2>,..)*
Ex:- Create Index NCPI_Eno_Phone ON Employee(Eno,Phone)

iii) **Unique index** :- Unique index doesn't allow duplicate values into the column on which index is created.
Syntax:- *CREATE UNIQUE INDEX <INDEX_NAME> ON <Table_Name> (<Column_Name>)*
Ex:- CREATE UNIQUE INDEX UQ_Phone ON Employee(Phone)

2) **Clustered Indexes** :- clustered index stores actual records and index & table are one. Only one CLUSTERED Index allowed per table. SQL Server creates clustered index on primary key column by default.
Syntax:- *Create Clustered Index <Index-Name> ON <Table-Name> (<Column-Name>)*
Ex:- CREATE CLUSTERED INDEX I10 ON CUST2(CID)

- When a Query send to sql server, it uses following methods to find desired record
1) **TABLE SCAN** :- Sql server scan each and every row if indexed column is not mentioned in Where clause. It is slow.
2) **Index SCAN** :- On an avg sql server scans only half of the table , so index scan is faster than table scan.

Reason of Index improve query performance
- When an indexes is created on column, internally a tree like structure is created in background that structured is called B-Tree Structure. In B-Tree Values of that indexed column are divided into two parts Left Side Nodes & Right side Nodes. Middle value taken as root node and value less then root node are arranged in left side & Value that are greater then Middle value are arranged into right side node. If your value is present greater than middle value of root node then search will be perform only Right side of Half Node and other Half Left side Node is ignored. So here searching time is half. This makes searching faster by using indexes.

Qns28. What is a materialized view?

Ans. Materialized view stores the actual data returned by the query unlike regular view, which is just stored query.

Advantage
1) **Performance Boost** :- Speeds up complex joins and aggregations by caching results
2) **Reduced Load** :- Minimizes repeated computation on base tables.
3) **Data is stored like a table.**
4) **Refreshable** :- Can be updated manually or automatically.
5) **Indexable**: You can create indexes for faster access.
6) **Queryable**: Acts like a regular table in SELECT statements.

- **SQL Server doesn't support Materialized view, but we can create same functionality using indexed view**
Ex:- CREATE VIEW GPAView WITH SCHEMABINDING AS SELECT s.StudentID, AVG(g.GradePoint) AS CGPA FROM Students s JOIN Grades g ON s.StudentID = g.StudentID GROUP BY s.StudentID;

- **Now, create a clustered index to materialize it :-**
Ex:- CREATE CLUSTERED INDEX idx_GPA ON GPAView(StudentID);

Qns29. What are transactions? Explain COMMIT, ROLLBACK, and SAVEPOINT.

Ans. - A transaction is a unit of work that contains one or more operations(I,U,D) and must be saved as a whole or must be cancelled as a whole.
- Every txn must gurantee a property called " **Atomicity** " i.e. all or none, if txn contains multiple operations then if all operations are successful then it must be saved , if one of the operation fails then entire txn must be cancelled.

Ex :- **Money Transfer** => Two Updates are performed [2 Operations]

Account 1----------------------------Money 1000--------------------------Account 2		
Update 1 (bal=bal-1000)	**Update 2 (bal=bal+1000)**	**Transaction**
Successful	**Failed**	Invalid
Failed	**Successful**	Invalid
Successful	**Successful**	Valid (Correct Txn)
Failed	**Failed**	valid (Correct Txn)

- The following commands provided by sql server to handle transactions called TCL commands.
COMMIT => Used to save the transaction permentanly in database memory.
ROLLBACK => Used to cancel the transaction if transaction begin using "begin transaction" explicitly.

- Every transaction has a "begin point" and an "end point".
Ex:- Go to ATM, Insert the card and Select "Money Withdrawal" => Begin Point
Next enter the Amount, then PIN, and Got the Cash =>End Point

- In sql server, a txn "begins implicitly with dml/ddl command" and "ends implicitly with commit".
- A user can also start txn by executing "begin transaction" command and ends transaction with commit/rollback. Due to this reason explicit transaction is recommended not implicit transaction.

```
Ex:- create table a(a int)
     begin transaction
        insert into a values(10)
        insert into a values(20)
        insert into a values(30)
        insert into a values(40)
        rollback
```

Note: **SAVEPOINT** word is not present in sql server, but similar function available with the name "**SAVE TRANSACTION**"

SAVE TRANSACTION => We can declare save transaction and we can rollback upto the save transaction. Using save transaction we can cancel part of the transaction.

```
Ex:- create table a(a int)
     begin transaction
        insert into a values(10)
        insert into a values(20)
     save transaction st1
         insert into a values(30)
         insert into a values(40)
     save transaction st2
         insert into a values(50)
         insert into a values(60)
     rollback transaction st2  => it will cancel the transaction upto st2 save point
```

Qns30. What are aggregate functions? List a few with examples

Ans. - Aggregate means, Getting one value from multiples values that is called Aggregate.
- These functions process multiple rows and returns one value

Aggregate Functions

1) **MAX( )** :- Returns Maximum value and it takes only one argument.
Syntax:- *MAX(Column_Name)*
Ex:- SELECT MAX(Sal) FROM Employee

2) **MIN( )** :- Returns Minimum value and it takes only one argument.
Syntax:- *MIN(Column_Name)*
Ex:- SELECT MIN(Sal) FROM Employee

3) **SUM( )** :- Returns total sum of a column values and it takes only one argument
Syntax:- *SUM(Column_Name)*
Ex:- SELECT SUM(Sal) AS TotSAL FROM Employee

4) **AVG( )** :- Returns average value of a Column and it takes only one argument
Syntax:- *AVG(Column_Name)*
Ex:- SELECT AVG(Sal) AS AverageSAL FROM Employee

NOTE => SUM( ) & AVG( ) functions cannot be applied on char, date columns.

5) **COUNT( )** :- Returns no of values present in a column and it takes only one argument. It doesn't count Null value.
Syntax:- *COUNT(Column_Name)*
Ex:- SELECT COUNT(Empno) AS No_Employees FROM Employee

6) **COUNT(*)** :- Returns no. of rows in a table. It count NULL value in a row
Syntax:- *COUNT(*)*
Ex:- SELECT COUNT(*) AS No_Employees FROM Employee

Qns31. How can you optimize a slow-running SQL query?

Ans. To optimize a slow-runnig SQL query, we need to take care and improve query as follow :-

1) Understand the Execution Plan
- Use "EXPLAIN" in (MySQL) or "SET SHOWPLAN_ALL ON" / "SET STATISTICS TIME ON" in (SQL Server)
- Observe Costly operations: Table Scan, Nested Loops, Missing Indexes

2) Indexing Strategy
- Add indexes on columns used in JOINS, WHERE, ORDER BY, GROUP BY.
- Use "covering indexes" to include all columns used in a query.
- Avoid over indexing, it slows down INSERT/UPDATE

3) Refactor Joins and Subqueries
- Replace correlated subqueries with JOINs where possible
- Use INNER JOIN instead of OUTER JOIN if nulls aren't needed
- Consider CTEs for readability and modularity

4) Filter Early

- User WHERE clauses on data source as possible
- Avoid functions on indexed columns in WHERE

5) Limit Rows and Columns

- Use "SELECT column1, column2" instead of "SELECT * "
- Apply "LIMIT" or "TOP" to reduce result set size during testing

6) Optimize Aggregations

- Use indexed columns in GROUP BY
- Use pre-aggregate data in materialized views or temp tables if reused often

7) Avoid Inefficient Patterns

- Replace IN (SELECT ...) with JOIN or EXISTS
- Avoid scalar functions in SELECT or WHERE => they run row-by-row

8) Partitioning

- Use table partitioning for large datasets

  EX:- semester-wise results

9) Caching and Temp Tables

- Cache frequent queries at the app level or use temp tables for intermediate results
- Use indexed temp tables for complex joins or aggregations

Qns32. What is the EXPLAIN or EXPLAIN PLAN statement used for?

Ans. The **EXPLAIN** or **EXPLAIN PLAN** statement is your window into the SQL engine's decision-making process. It shows you how a query will be executed—step by step—before it actually runs. This is essential for diagnosing performance issues and optimizing queries.

Qns33. How does indexing affect INSERT, UPDATE, and DELETE performance?

Ans. <u>Impact on INSERT</u>

- Every time we insert a row, the database must update all relevant indexes.
- More indexes = more overhead.
- Clustered indexes may cause page splits if the insert interrupt physical order.
- Non-clustered indexes require separate updates to maintain index structure.
- Result : Slower inserts, especially in bulk operations or high-volume tables.

<u>Impact on UPDATE</u>

- If you're updating a column that's part of an index, the index must be modified.
- This can involve removing and reinserting entries in the index.
- If the column isn't indexed, the impact is minimal.

Impact on DELETE

- Deletes must remove the row from the table and from all indexes that reference it.
- If the WHERE clause uses an index, it can speed up locating rows to delete.
- But once found, every index entry must be cleaned up.
- Result : Faster targeting, but slower cleanup if many indexes exist.

Qns34. What is a composite index and when should it be used?

Ans. - IF index created on Muliple columns then it is called Composite index.
- It's designed to optimize queries that filter, join, or sort using multiple columns together

When to Use

- Columns Are Frequently Used Together.
  Ex:- "student_id" and "semester" column often appear together in WHERE, JOIN, or ORDER BY, a composite index is ideal.

Qns35. What is normalization overhead and how do you deal with it?

Ans. Normalization improves data integrity and reduces redundancy by splitting data into multiple related tables. But this comes at a cost:

1) Increased Joins

- Queries often require multiple JOINs to inorder to get required data
- More joins = more CPU and memory

2) Complex Query Logic

- Writing and maintaining queries becomes harder
- Debugging and optimizing multi-table joins takes more time

How to Deal

1) Use Denormalization Selectively

- Create summary tables or materialized views for frequent queries
  Ex:- semester-wise CGPA or rank lists

2) Index Strategically

- By adding composite indexes on foreign keys and join columns
- Use covering indexes for read-heavy queries

3) Optimize Joins

- Use INNER JOIN where possible
- Filter early using indexed columns
- Avoid joining unnecessary tables

4) Use Stored Procedures

- Write complex logic inside Stored Procedure to reduce query parsing overhead

Qns36. How do you avoid Cartesian products in JOINs?

Ans. Avoiding Cartesian products in SQL JOINs is all about ensuring that your tables are joined with explicit, meaningful conditions. A Cartesian product happens when every row from one table is matched with every row from another, which can explode result set and kill performance.

Way to avoid

- Use INNER JOIN, LEFT JOIN, etc., with clear ON conditions
- Verify Join Conditions by taking care or joining table on related key
  Ex:- Foreign Key <--> Primary Key
- Use Alias help clarify which table each column belongs to, reducing mistakes.

Qns37. What is partitioning in SQL?

Ans. Partitioning in SQL is a technique used to divide large tables or indexes into smaller, more manageable pieces, called partitions, while still treating them as a single logical entity.

Ex:- It's especially useful in systems academic data (e.g., results, enrollments, CGPA records) can grow semester by semester or year by year

Types of Partitioning

1) Horizontal Partitioning (Table Partitioning) :- Splits a table row-wise based on a column value
Ex:- semester, year, campus

2) Vertical Partitioning :- Splits a table column-wise into multiple tables
Ex:- students_basic (student_id, name, dob)
students_sensitive (student_id, aadhaar, address)

3) Partitioning in Window Functions :- Used with PARTITION BY inside OVER() clause to segment data for analytics

Ex:-
```
SELECT student_id, semester,
RANK() OVER (PARTITION BY semester ORDER BY GPA DESC) AS rank
FROM GPA;
```

Qns38. What causes a deadlock in SQL, and how can you prevent it?

Ans. A deadlock in SQL occurs when two or more transactions are waiting on each other to release locks, creating a cycle where none can proceed.

Cause of Deadlock

1) Mutual Exclusion :- A resource (e.g., row, table) is locked by one transaction at a time
2) Hold and Wait :- A transaction holds one lock and waits for another.
3) Circular Wait :- A cycle of transactions each waiting for a resource held by the next

Ex:- * Transaction A locks "Students" table and wants data from "Grades" table
* Transaction B locks "Grades" table and wants "Students" table
* Both wait on each other -> deadlock

Prevention from Deadlock

1) Access Resources in a Consistent Order

- Always lock tables in the same sequence across transactions.

Ex:- Always access "Students" Table -> "Enrollments" Table -> "Grades" Table

2) Keep Transactions Short and Focused

- Minimize the time locks are held. Avoid user input or long logic inside transactions.

3) Avoid Nested Transactions and Triggers

- Complex triggers or nested procedures can unintentionally create circular waits.

Qns39. What is the difference between clustered and non-clustered indexes?

Ans.

Non clustered	Clustered
1) Stores pointers to actual records	1) Stores actual records
2) Table and index are separate objects	2) Table and index are not separate
3) Needs extra storage	3) Doesn't need extra storage
4) SQL server requires two lookups to find desired record	4) SQL server requires only one lookup
5) SQL server requires only one lookup per table	5) Only one clustered index allowed per table
6) By default it is created on unique column	6) By default it is created on pk column

Qns40. What tools do you use to monitor SQL query performance?

Ans. I used SQL Server Management Studio (SSMS).

Qns41. You are asked to design a student-course grading system. What tables and relationships would you create?

Ans. The Student Result Processing System consists of 6 core Tables

1) **Students** - Core student information
2) **Programme** - Academic programs/courses
3) **Semesters** - Academic semesters
4) **Subject** - Course subjects
5) **Grades** - Individual subject grades
6) **GPA** - Semester-wise GPA calculations

Relationship among the Tables

Relationship	Cardinality	Description
Students → Programme	Many-to-One	Many student belongs to one programme
Students → Grades	One-to-Many	Each student can have multiple grades
Students → GPA	One-to-Many	Each student can have multiple semester GPAs
Programme → Subject	One-to-Many	Each programme contains multiple subjects
Semesters → Subject	One-to-Many	Each semester offers multiple subjects
Subject → Grades	One-to-Many	Each subject can have multiple student grades
Semesters → GPA	One-to-Many	Each semester can have multiple student GPAs

Qns44. What would you do if a production database is missing some records due to a failed update?

Ans. I can take following steps:

1) Stop Further Writes

- Temporarily disable write operations to prevent further data corruption.
- Alert relevant teams if this impacts live systems.

2) Check Transaction Logs

- Use tools like ApexSQL Log (SQL Server) to read transaction logs and identify what was updated or deleted.
- If I may be able to roll back the failed update if the logs are intact.

3) Restore from Backup

- If logs aren't sufficient, restore the most recent backup to a staging environment.
- Compare the staging data with production to identify missing records.

4) Reconstruct Missing Data

- Use the backup or audit tables to manually reinsert or update lost records.
- Validate against business rules (e.g., GPA formulas, enrollment constraints).

5) Audit and Log Everything

- If not already in place, implement before-update triggers or change tracking to log future modifications.

Qns45. How would you implement role-based access to sensitive information in SQL?

Ans. To implement role-based access we need to follow below steps:

1) Define Roles Based on Responsibilities

Ex:- In an academic system:

i) **Admin** :- Full access to all tables and procedures
ii) **Faculty** :- Can view and update grades for their subjects
iii) **Student** :- Can view their own results only
iv) **Auditor** :- Read-only access to all academic records

2) Create Roles in SQL Server

Ex:- *CREATE ROLE Faculty;*
CREATE ROLE Student;
CREATE ROLE Admin;

3) Assign Permissions to Roles

- Write below command in SQL Server

Ex:- *GRANT SELECT, UPDATE ON Grades TO Faculty;*
GRANT SELECT ON Results TO Student;
*GRANT ALL ON *.* TO Admin;*

- Use *GRANT EXECUTE* for stored procedures if logic is encapsulated.

4) Create Users and Assign Roles

- Write below command

Ex:- *CREATE USER prof_Raju FOR LOGIN prof_Raju_login;*
ALTER ROLE Faculty ADD MEMBER prof_Raju;

5) Use Views or Stored Procedures for more Control

Ex:- **restrict students to only their own data:**

CREATE VIEW StudentResults AS SELECT student_id, semester, GPA
FROM GPA WHERE student_id = SYSTEM_USER(); -- or SESSION_USER()

- If encapsulate logic inside stored procedure :

Ex:-
```
CREATE PROCEDURE GetStudentResults(IN sid INT)
BEGIN
    SELECT * FROM Results WHERE student_id = sid;
END;
```

- Then,

EX:- *GRANT EXECUTE ON GetStudentResults TO Student;*

Qns46. You are given a raw CSV with dirty data. How would you load and clean it using SQL?

Ans. Cleaning a raw CSV with dirty data using SQL is a multi-step process that includes data ingestion, validation, and transformation. To do this follow the below steps:

Step 1:- Load the CSV into a Staging Table

1) Create a flexible staging table in SQL:- Use VARCHAR for all columns to avoid type errors during import.

Ex:-
```
CREATE TABLE staging_raw_data
(
    student_id VARCHAR(50),
    name VARCHAR(100),
    dob VARCHAR(20),
    semester VARCHAR(20),
    gpa VARCHAR(10)
);
```

Step 2: Load the CSV to use bulk import tools in SQL Server "BULK INSERT" or "OPENROWSET".

Ex:-
```
BULK INSERT staging_raw_data
FROM 'C:\path\to\file.csv'
WITH ( FIELDTERMINATOR = ',' , ROWTERMINATOR = '\n', FIRSTROW = 2 );
```

Step 3 : Clean the Data in SQL

- Remove Duplicates

Ex:-
```
DELETE FROM staging_raw_data
WHERE student_id IN (
    SELECT student_id
    FROM staging_raw_data
    GROUP BY student_id
    HAVING COUNT(*) > 1
);
```

- Normalize Dates

Ex:-
```
UPDATE staging_raw_data SET dob = TRY_CONVERT(DATE, dob);
```

- Validate GPA Format

Ex:-
```
UPDATE staging_raw_data SET gpa = NULL
WHERE TRY_CAST(gpa AS DECIMAL(4,2)) IS NULL OR gpa NOT BETWEEN 0 AND 10;
```

- Trim and Standardize Text

Ex:-
```
UPDATE staging_raw_data SET name = TRIM(name),
semester = UPPER(TRIM(semester));
```

Step 4: Move Clean Data to Final Table

- Create a strongly typed table:

EX:-
```
CREATE TABLE students_clean
(
    student_id INT PRIMARY KEY,
    name VARCHAR(100),
    dob DATE,
    semester VARCHAR(20),
    gpa DECIMAL(4,2)
);
```

- Insert validated rows:

Ex:-
```
INSERT INTO students_clean (student_id, name, dob, semester, cgpa)
SELECT
    TRY_CAST(student_id AS INT),name, TRY_CAST(dob AS DATE), semester,
    TRY_CAST(gpa AS DECIMAL(4,2)) FROM staging_raw_data
    WHERE TRY_CAST(student_id AS INT) IS NOT NULL
    AND TRY_CAST(gpa AS DECIMAL(4,2)) BETWEEN 0 AND 10;
```

Qns48. What measures would you take to secure a database with sensitive data?

Ans. To secure a database with sensitive data, I will take following below measures

1) Role-Based Access Control (RBAC)
2) Data Encryption
3) Strong Authentication

4) Input Validation & Parameterized Queries
5) Data Masking
6) Audit Logging
7) Regular Backups & Point-in-Time Recovery

Qns49. How do you create daily backup and restore plans for a SQL database?

Ans. I will create a stored procedure which takes full backup's of all databases and stored at particular location with the file name as database name concatenate with date on which backup taken and to automate the process I will create a Trigger which will run the procedure at particular time.

Ex:- **Creating stored procedure to take backup**

```
CREATE OR ALTER PROCEDURE SP_Backup_Databases
AS
Begin
  DECLARE C1 CURSOR FOR SELECT name FROM SYS.DATABASES WHERE database_id>4
  DECLARE @dbname varchar(30), @fname varchar(40)
  OPEN C1
  FETCH NEXT FROM C1 INTO @dbname
  WHILE(@@FETCH_STATUS=0)
  BEGIN
    SET @fname='E:\Files\'+@dbname+FORMAT(GETDATE(),'yyyyMMdd')+'.bak'
    BACKUP DATABASE @dbname TO DISK=@fname
    FETCH NEXT FROM C1 INTO @dbname
  END
  CLOSE C1
  DEALLOCATE C1
End
```

- **Executing procedure**

Ex:- *Execute SP_Backup_Databases*