1. **Constraints in SQL**NOT NULL, CHECK, DEFAULT, UNIQUE, INDEX, PRIMARY KEY, FOREIGN KEY
2. **SQL Join & its Types**In SQL, a join is used to combine data from two or more tables in a relational database based on a related column between them. (**Use Venn Diagram to understand the joins**)There are several types of joins, including:
3. **INNER JOIN**: Returns only the matching rows between two tables. It combines rows from two tables based on a specified condition. Trick -> make a Cartesian product and apply condition.

*Example:*

*Let's say we have two tables: Orders and Customers. We want to retrieve the orders made by customers. An inner join would look like this:*

*SELECT Orders.OrderID, Customers.CustomerName*

*FROM Orders INNER JOIN Customers*

*ON Orders.CustomerID = Customers.CustomerID;*

*This query will return the OrderID and CustomerName for only the orders that have a matching customer in the Customers table.  
Note: We can also write* ***JOIN*** *instead of* ***INNER JOIN****. JOIN is same as INNER JOIN.*

1. **LEFT JOIN**: Returns all the rows from the left table and matching rows from the right table. If there are no matching rows in the right table, the result will still contain all the rows from the left table.  
   *If you want to retrieve a list of all customers and their orders (including customers with no orders), a full outer join could be used like this:*

*SELECT Customers.CustomerName, Orders.OrderID*

*FROM Customers FULL OUTER JOIN*

*Orders ON Customers.CustomerID = Orders.CustomerID;*Note: We can also use **LEFT OUTER JOIN** instead of **LEFT JOIN**, both are the same.

1. **RIGHT JOIN**: Returns all the rows from the right table and matching rows from the left table. If there are no matching rows in the left table, the result will still contain all the rows from the right table.  
   *To retrieve a list of all orders and their corresponding customers (including orders with no associated customers), a right join can be used:*

*SELECT Customers.CustomerName, Orders.OrderID*

*FROM Customers RIGHT JOIN Orders*

*ON Customers.CustomerID = Orders.CustomerID;*

*Note: We can also use* ***RIGHT OUTER JOIN*** *instead of* ***RIGHT JOIN****, both are the same.*

1. **FULL OUTER JOIN**: Returns all the rows from both tables, with NULL values in the columns where there are no matches. It is combination of left join and right join.  
   Example:

If you want to retrieve a list of all customers and their orders (including customers with no orders), a full outer join could be used like this:

SELECT Customers.CustomerName, Orders.OrderID

FROM Customers FULL OUTER JOIN Orders

ON Customers.CustomerID = Orders.CustomerID;

1. **CROSS JOIN**: Returns the cartesian product of both tables, which means that every row from the first table will be combined with every row from the second table.  
   *SELECT table1.column1 , table1.column2, table2.column1...*

*FROM table1* ***CROSS JOIN*** *table2;*

*table1: First table. table2: Second table*

1. **SELF JOIN**: A join where a table is joined with itself, typically used when there is a hierarchical relationship within the table.  
   *SELECT a.coulmn1 , b.column2*

*FROM table\_name a, table\_name b*

*WHERE some\_condition;*

*table\_name: Name of the table. some\_condition: Condition for selecting the rows.*

1. In addition to these basic types of joins, there are also some advanced types of joins such as **natural join, equijoin, and non-equijoin**, but these are less commonly used.

Example Present on https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/ & https://www.geeksforgeeks.org/sql-join-cartesian-join-self-join/

1. **What are Indexes and how to create an index in SQL.**

In SQL, indexes are database objects that are used to improve the performance of queries by providing faster access to data. An index is essentially a data structure that allows you to quickly look up rows based on the values of one or more columns.

When you create an index on a table, the database engine creates a separate data structure that stores a sorted copy of the column or columns being indexed, along with a pointer to the corresponding row in the table. This allows the database engine to quickly find the rows that match a query condition that references the indexed column(s).

Indexes can be created on one or more columns of a table, and can be either unique or non-unique. Unique indexes enforce a constraint that ensures that the values in the indexed column(s) are unique across all rows in the table.

While indexes can significantly improve query performance, they do have some trade-offs. Creating and maintaining indexes can consume additional storage space and can slow down data modification operations (such as inserts, updates, and deletes) because the database engine needs to update the index as well as the table. Additionally, using too many indexes on a table can actually slow down query performance, as the database engine needs to spend more time searching and comparing index entries.

Syntax: *Create Index <Index-name> on TABLE\_NAME(col1, col2);*  
Syntax to drop index: *DROP INDEX <Index-name> on TABLE\_NAME;  
Link to video:* [*https://www.youtube.com/watch?v=FJdryrcDD2s*](https://www.youtube.com/watch?v=FJdryrcDD2s)

1. Partioning
2. Mapping
3. **Foreign Key:-** It is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table. The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table. Primary table m agar kuch delete kiya toh foreign key wale m se apne aap delete ho jayega. Issi ko referential integrity bolte hai.
4. **Different Types Of SQL Commands**
5. Data Definition Language:- Create, Alter, Drop, Truncate, Rename
6. Data Manipulation Language: - Select, Insert, Update, and Delete.
7. Data Control Language: - Grant, Revoke.
8. Transaction Control Language: -Commit, Rollback, Savepoint.
9. **Delete v/s Drop v/s Truncate**

|  |  |  |
| --- | --- | --- |
| Delete | Drop | Truncate |
| Removes rows from a table | Remove a table from DB | Removes all rows from table |
| DML | DDL | DDL |
| Can be rolled back | Can’t be rolled back | Can’t be rolled back |

1. **Group By v/s Order By**  
   Group By is applicable when we want to use aggregate function to more than one set of rows.  
   Order By is applicable when we want to get data obtained by query in the sorting order.
2. **Pattern Matching**

Where Name Like ‘a%’, Where Name Like ‘%a’, Where Name Like ‘%a%’, Where Name Like ‘\_a%’, Where Name Like ‘a\_%’, Where Name Like ‘a%y’.

1. **Triggers: -** It is special type of stored procedure that automatically runs when an event occurs in database server.
2. **ACID Properties: -**
3. **Atomicity:** Either all the changes are performed, or none of them.
4. **Consistency:** Data is in consistent state when a transaction starts and when it ends
5. **Isolation:** Transactions that runs currently appears to be serialised.
6. **Durability:** After atransactionis successfully completed, data will persist in DB, even if the system fails.
7. **What are alias?**

Aliases are used to give a table, or a column in a table, a temporary name.

An alias is created with the AS keyword.

Example:

SELECT column\_name AS alias\_name FROM table\_name;

1. **What are stored procedures and how to create them.**

Stored procedure is an SQL code which can be saved and reused again and again.

Example in mySql  
Creating a Stored Procedure

CREATE PROCEDURE GetCustomerOrders(IN customerID INT)

BEGIN

SELECT \* FROM orders WHERE customer\_id = customerID;

END;

Calling a stored procedure.

CALL GetCustomerOrders(1);

1. **What is View? How can we create Views?**

View is a virtual table, meaning it doesn't store data but retrieves data dynamically from the underlying tables when queried.

Views can join multiple tables, filter data, and define a subset of columns, presenting a simplified and customized view of the data.

Creating a view.

CREATE VIEW employee\_info AS

SELECT e.employee\_id, e.first\_name, e.last\_name, d.department\_name

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id;

Calling a view

SELECT \* FROM employee\_info;

1. **What are window function in sql?**

Window: It is a set of rows. In a table we can have more than 1 window.

Windows functions are SQL functions that enable us to perform operations on a window – i.e. a set of records.

Window is created using the OVER() clause in SQL.

Over clause does following things:

* Partitioning (using Partition By): It divides the rows into partitions, so we can specify which rows to use.
* Ordering (using Order By): It orders the rows within each partition. This is optional.
* Rows: It can be used to further limit the rows within our partition. This is optional.

Different Types of Window Functions

There are a lot of window functions that exist in SQL but they are primarily categorized into 3 different types:

* **Aggregate window functions**: Aggregate window functions are used to perform operations on sets of rows in a window(s). They include SUM(), AVG(), MIN(), MAX(), COUNT(), etc.
* **Value/Analytic window functions**: Value window functions are like aggregate window functions that perform multiple operations in a window, but they're different from aggregate functions. They include things like LAG(), LEAD(), FIRST\_VALUE(),LAST\_VALUE(), etc.
* **Ranking window functions**: Rank window functions are used to rank rows in a window(s). They include RANK(), DENSE\_RANK(), ROW\_NUMBER(), PERCENT\_RANK(), RANK(), etc.

SQL Questions

1. Find third highest salary or Nth highest salary

Third Highest Salary : Select salary FROM Emp\_tbl ORDER BY Salary DESC LIMIT 2,1;  
Nth Highest Salary : Select salary FROM Emp\_tbl ORDER BY Salary DESC LIMIT N-1,1;

Here N-1 is starting index and 1 is number of rows to fetch.

1. Finding Max Salary In Each Department  
   Select dept\_name, max(sal) from emp\_tbl GROUP BY dept\_name;
2. Finding Min Salary In Each Department  
   Select dept\_name, min(sal) from emp\_tbl GROUP BY dept\_name;
3. Find Number of employees in each department

Select dept\_name, Count(\*) from emp\_tbl GROUP BY dept\_name;

1. Display alternate records in MySql

Select emp\_number, emp\_name, salary, rownum rn from where mod(rn,2)=1

1. Find duplicated values and its frequency of a column.  
   Select emp\_name, count(\*) from emp\_tbl GROUP BY emp\_name HAVING count(\*)>1;
2. Display name of employees whose name starts with ‘M’  
   Select emp\_name from emp\_tbl where emp\_name LIKE ‘M%’;
3. Display name of employees whose name end with ‘N’  
   Select emp\_name from emp\_tbl where emp\_name LIKE ‘%N’;
4. Display name of employees whose name contains letter ‘M’  
   Select emp\_name from emp\_tbl where emp\_name LIKE ‘%M%’
5. Display name of employees whose name does not contains letter ‘M’  
   Select emp\_name from emp\_tbl where emp\_name NOT LIKE ‘%M%’