**Basic Level (Beginner)**

**1.**

**2.**

**3.**

**4.**

**5.**

**6.**

**7.**

**8.**

**9.**

**10.**

**11.**

**12.**

**13.**

**14.**

**15.**

**16.**

**17.**

**18.**

**19.**

**20.**

**21.**

**22.**

**23.**

**24.**

**25.**

**What is SQL?**

**What is a Database?**

**What are the types of SQL commands? What is Primary Key?**

**What is Foreign Key? What is UNIQUE Key?**

**What is the difference between Primary Key and UNIQUE Key? What is NOT NULL constraint?**

**What is Default Constraint?**

**What is the difference between DELETE, TRUNCATE, and DROP? What is the difference between WHERE and HAVING?**

**What are Joins in SQL? What is INNER JOIN? What is LEFT JOIN?**

**What is RIGHT JOIN? What is FULL JOIN? What is Self Join?**

**What is Cross Join?**

**What is Union and Union All?**

**What is the difference between UNION and UNION ALL? What is Normalization?**

**What is Denormalization?**

**What is the difference between CHAR and VARCHAR? What is the difference between SQL and MySQL?**

**What is Auto Increment in SQL?**

**Intermediate Level**

**26.**

**27.**

**28.**

**29.**

**30.**

**31.**

**32.**

**33.**

**34.**

**35.**

**36.**

**37.**

**38.**

**39.**

**40.**

**41.**

**42.**

**43.**

**44.**

**45.**

**46.**

**47.**

**48.**

**49.**

**50.**

**What is Subquery?**

**What is Nested Query?**

**What is Correlated Subquery? What is Group By in SQL?**

**What is the difference between Group By and Order By? What is the use of LIMIT in SQL?**

**How to find the Second Highest Salary in SQL? How to find Duplicate Records in a table?**

**What is CTE (Common Table Expression)? What is Temporary Table in SQL?**

**What is Window Function in SQL?**

**What is the difference between ROW\_NUMBER(), RANK(), and DENSE\_RANK()?**

**What is CASE Statement in SQL? What is COALESCE in SQL?**

**What is NVL Function in SQL? What is Indexing in SQL?**

**What is Clustered Index? What is Non-Clustered Index?**

**What is the difference between Clustered and Non-Clustered Index? What is View in SQL?**

**What is the difference between View and Table? What is Stored Procedure?**

**What is the difference between Function and Stored Procedure? What is Trigger in SQL?**

**What is Cursor in SQL?**

**Advanced Level**

**51.**

**52.**

**53.**

**54.**

**55.**

**56.**

**57.**

**58.**

**59.**

**60.**

**61.**

**62.**

**63.**

**64.**

**65.**

**66.**

**67.**

**68.**

**69.**

**70.**

**71.**

**72.**

**73.**

**74.**

**75.**

**What is the ACID Property in SQL?**

**What is a Transaction in SQL?**

**What is the difference between COMMIT and ROLLBACK? What is Savepoint in SQL?**

**What is the difference between IN and EXISTS?**

**What is the difference between DELETE and TRUNCATE? What is Index Fragmentation?**

**What is the difference between RANK() and DENSE\_RANK()? How to fetch common records from two tables?**

**What is the difference between UNION and JOIN? What is Pivot Table in SQL?**

**What is Case Sensitivity in SQL? How to find the Nth Highest Salary?**

**How to get First 3 Maximum Salaries?**

**What is the difference between Drop, Delete, and Truncate? How to calculate Age from Date of Birth in SQL?**

**What is Recursive Query in SQL?**

**What is the difference between Temporary Table and CTE? How to find Odd and Even records in SQL?**

**What is JSON in SQL? What is XML in SQL?**

**How to handle NULL values in SQL? What is Dynamic SQL?**

**How to calculate Percentage in SQL?**

**How to find the Employees who earn more than their Manager?**

**Real-Time Scenarios**

**76.**

**77.**

**78.**

**79.**

**80.**

**81.**

**82.**

**83.**

**84.**

**85.**

**86.**

**87.**

**88.**

**89.**

**90.**

**How to find Duplicate Emails in the Employee Table?**

**How to get the Highest Salary in each Department? How to find Employees joined in the last 3 months?**

**How to Display the First 5 Records in SQL?**

**How to find the Number of Employees in each Department?**

**How to find the Last 3 Records in SQL? How to find Employees without Managers? How to find the First Name starting with 'A'? How to fetch Alternate Rows from a table?**

**How to swap two columns in SQL?**

**How to display the Duplicate Records with their Count? How to find the Highest Salary without using MAX()?**

**How to fetch common records from two tables without JOIN?**

**How to delete Duplicate Records from a table?**

**How to find the Department with the highest Employee Count?**

**Optimization Techniques**

**91.**

**92.**

**93.**

**94.**

**95.**

**96.**

**97.**

**98.**

**99.**

**100.**

**How to Optimize SQL Queries?**

**What is Query Execution Plan?**

**How to Improve Query Performance? What is Indexing?**

**What is Table Partitioning?**

**How to Avoid Deadlocks in SQL? What is the use of EXISTS in SQL? What is Query Optimization?**

**What is the Difference Between Stored Procedure and Function in**

**SQL?**

**What is the difference between OLTP and OLAP in SQL?**

**Basic Level (Beginner)**

## What is SQL?

SQL (Structured Query Language) is a standard programming language used to interact with relational databases. It is used to store, retrieve, update, and delete data. SQL is also used to create and modify database structures such as tables, views, and indexes.

# Example:

### SELECT \* FROM Employees;

This query retrieves all the records from the Employees table.

## What is a Database?

A database is an organized collection of data that is stored and managed electronically. It allows users to efficiently store, retrieve, update, and manage data. Databases are used to handle large amounts of information in various applications such as websites, business systems, and applications.

# Example:

A customer database in an e-commerce website may store customer details like name, email, contact number, and purchase history.

## What are the types of SQL commands?

SQL commands are categorized into five types based on their functionality:

* 1. **DDL (Data Definition Language)** – Defines the structure of the database.

 CREATE, ALTER, DROP, TRUNCATE

* 1. **DML (Data Manipulation Language)** – Manages data stored in the database.

 SELECT, INSERT, UPDATE, DELETE

* 1. **DCL (Data Control Language)** – Controls access to the data.

 GRANT, REVOKE

* 1. **TCL (Transaction Control Language)** – Manages transactions in the database.

 COMMIT, ROLLBACK, SAVEPOINT

* 1. **DQL (Data Query Language)** – Retrieves data from the database.
     + SELECT

## What is Primary Key?

A Primary Key is a column or a combination of columns in a table that uniquely identifies each row in that table. It does not allow NULL values and must always contain unique values.

### Key Features:

* Uniquely identifies each record
* Cannot have duplicate values
* Cannot contain NULL values
* Only one primary key is allowed per table

### Example:

Here, EmployeeID is the primary key that uniquely identifies each employee.

## What is Foreign Key?

A Foreign Key is a column or combination of columns in one table that refers to the Primary Key in another table. It is used to create a relationship between two tables and enforce referential integrity.

### Key Features:

* Establishes a relationship between two tables
* Can contain duplicate values
* Can accept NULL values
* Helps maintain data consistency

### Example:

Here, DepartmentID in the Employees table is a foreign key that references the DepartmentID column in the Departments table.

## What is UNIQUE Key?

A UNIQUE Key is a constraint that ensures all values in a column or combination of columns are distinct across all rows in the table. It prevents duplicate values but allows NULL values (only one NULL value in most databases).

### Key Features:

* Ensures uniqueness of each record in the column
* Allows one NULL value (depending on the database)
* Multiple UNIQUE keys can be defined in a table
* Helps maintain data integrity

**Example:**

Here, the Email column has a UNIQUE constraint, ensuring no two employees can have the same email address.

## What is the Difference Between Primary Key and UNIQUE Key?

**Primary Key**

Uniquely identifies each row in a table

Does not allow NULL values

Only one primary key is allowed per table

Automatically creates a unique clustered index

Used to uniquely identify a record

**UNIQUE Key**

Ensures all values in the column are unique

Allows one NULL value (in most databases)

Multiple UNIQUE keys can be defined in a table

Creates a unique non-clustered index

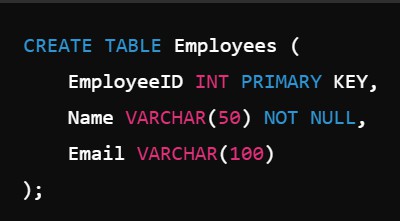
Used to enforce uniqueness in a column without being a primary identifier

1. **What is NOT NULL Constraint?**

The NOT NULL constraint ensures that a column cannot have NULL values. It is used to enforce that every row must have a value in that column.

**Key Features:**

* Prevents insertion of NULL values
* Ensures mandatory fields have data
* Can be applied to one or more columns

**Example:**

In this example, the Name column cannot have NULL values, while the Email column can accept NULL values.

## What is Default Constraint?

The Default Constraint provides a default value for a column when no value is specified during the insertion of a new record.

### Key Features:

* Automatically assigns a default value if no value is provided
* Helps avoid NULL values in specific columns
* Can be applied to any data type

### Example:

In this example, if no salary is provided while inserting a record, the Salary column will automatically be set to 5000.

## What is the Difference Between DELETE, TRUNCATE, and DROP?

**Command**

**DELETE**

**TRUNCATE**

**DROP**

**Function**

Removes specific rows based on a condition using the WHERE

clause

Removes all rows from the table without a condition

Deletes the entire table including data and structure

**Can Rollback**

Yes (with COMMIT/RO LLBACK)

No

No

**Affects Structure**

No

No

Yes (Removes table structure)

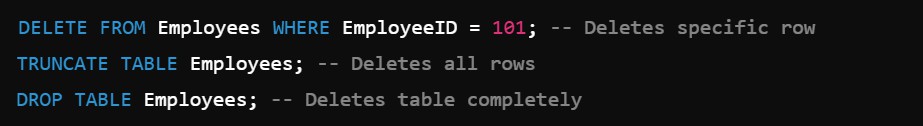
**Speed**

Slow (Row-by- row deletion)

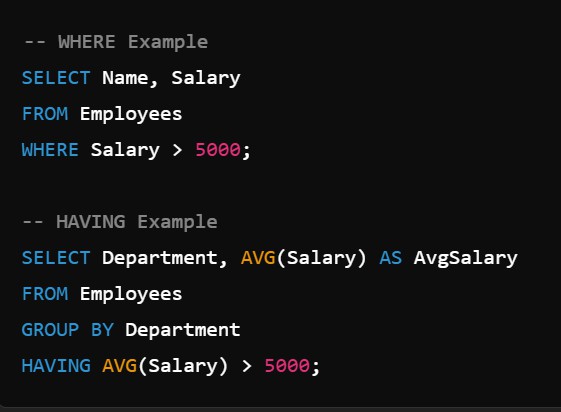
Faster than DELETE

Fastest

### Example:

****

**11. What is the Difference Between WHERE and HAVING?**

**Example:**

**Clause**

**WHERE**

**HAVING**

**Purpose**

Filters rows before grouping

Filters groups after grouping

**Used With**

SELECT, UPDATE, DELETE

SELECT with GROUP BY

**Filter Type**

Row-level filter

Group-level filter

**Execution Order**

Applied before GROUP BY

Applied after GROUP BY

In the WHERE clause, filtering is applied before grouping, while HAVING filters the aggregated result.

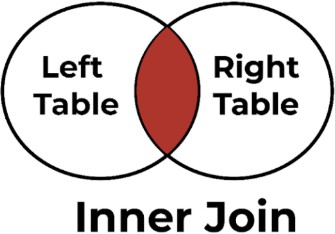
## What are Joins in SQL?

Joins in SQL are used to combine data from two or more tables based on a related column between them.

### Types of Joins:

* 1. **INNER JOIN** – Returns only matching rows from both tables.
  2. **LEFT JOIN** – Returns all rows from the left table and matching rows from the right table.
  3. **RIGHT JOIN** – Returns all rows from the right table and matching rows from the left table.
  4. **FULL JOIN** – Returns all rows from both tables (matching and non-matching).
  5. **SELF JOIN** – Joins a table with itself.
  6. **CROSS JOIN** – Returns the Cartesian product of both tables (all possible combinations).

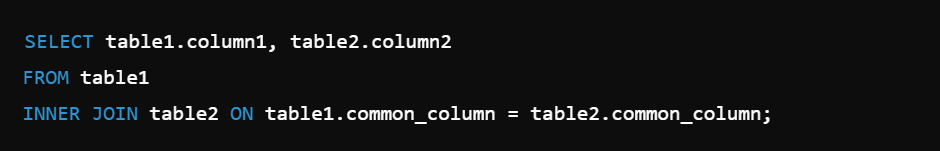
## What is INNER JOIN?

INNER JOIN is used to combine rows from two or more tables based on a matching condition between them. It returns only those records where the specified condition is true in both tables.

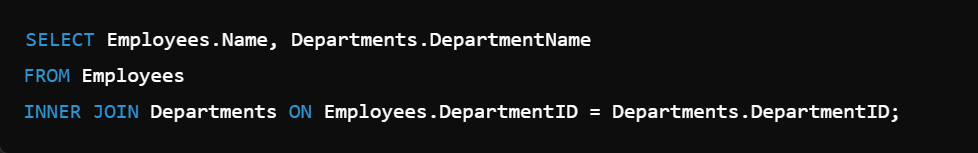
### Key Features:

* Returns matching rows from both tables
* Ignores unmatched rows
* Most commonly used type of join

### Syntax:

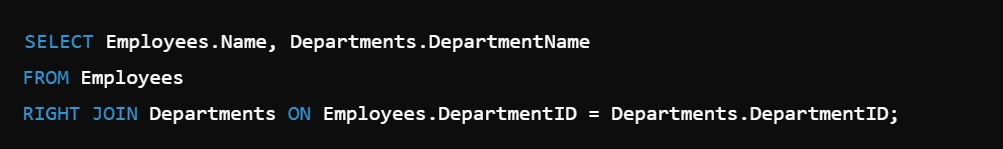
****

**Example:**

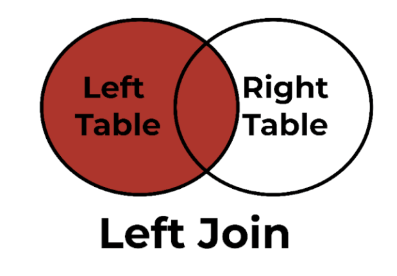
****

This query returns the employee names along with their department names where the DepartmentID is common in both tables.

## What is LEFT JOIN?



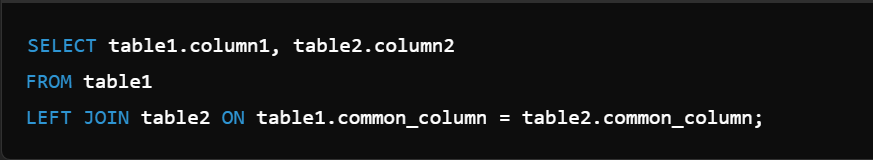
This query returns all department names from the Departments table, and if no employee is assigned to a department, the Employee Name will be displayed as NULL.

LEFT JOIN is used to return all records from the left table and the matching records from the right table. If no match is found, the result will contain NULL values from the right table.

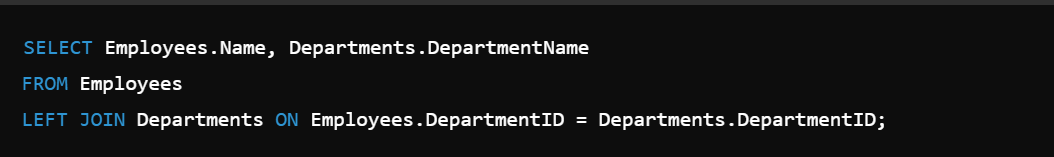
### Key Features:

* Returns all rows from the left table
* Returns matching rows from the right table
* Displays NULL for non-matching rows from the right table

### Syntax:

****

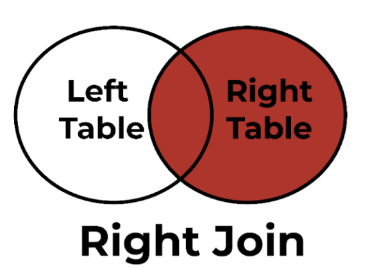
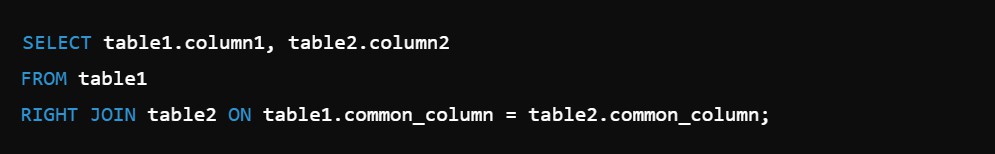
**Example:**



This query returns all employee names from the Employees table, and if the department is not assigned, the Department Name will be displayed as NULL.

## What is RIGHT JOIN?

RIGHT JOIN is used to return all records from the right table and the matching records from the



left table. If no match is found, the result will contain NULL values from the left table.

**Key Features:**

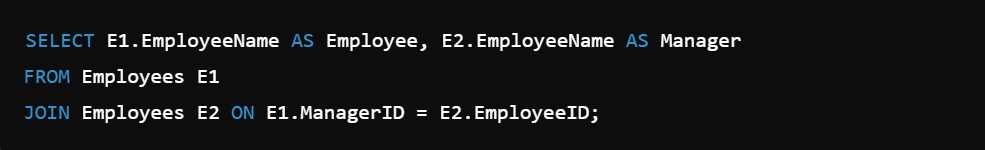
Returns all rows from the right table Returns matching rows from the left table

Displays NULL for non-matching rows from the left table

**Syntax:**

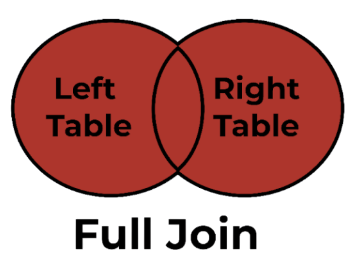
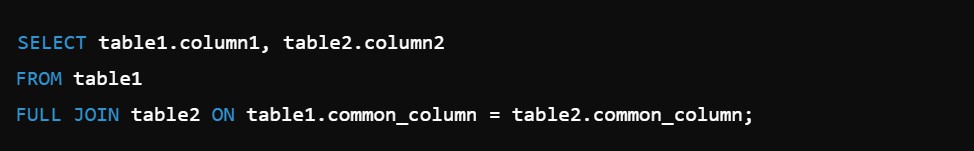
**Example:**

## What is FULL JOIN?



In this example, the Employees table joins with itself to show the employee's name along with their manager's name based on the ManagerID column.

FULL JOIN combines the results of both LEFT JOIN and RIGHT JOIN. It returns all records from both tables, with matching rows from both sides where available. If there is no match, the result will contain NULL values on the side where no match was found.



**Key Features:**

Returns all rows from both tables

* Displays NULL where there is no match
* Useful to find unmatched records in both tables

#### Syntax:

**Example:**

This query returns all employee names and department names, including those where there is no matching DepartmentID between the two tables.

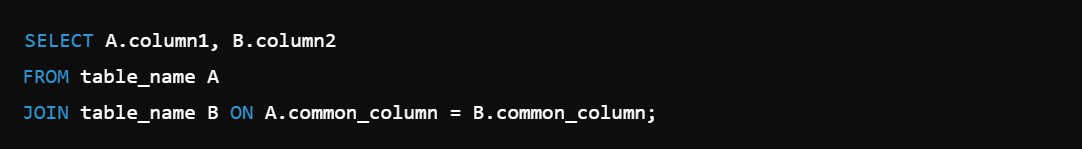
## What is Self Join?

Self Join is a type of join where a table is joined with itself to compare rows within the same table. It is used when a table contains a hierarchical relationship or when comparing values in the same table.

#### Key Features:

* Joins a table with itself
* Requires table aliases to differentiate table instances
* Used to compare rows within the same table

**Syntax:**

****

**Example:**

## What is Cross Join?



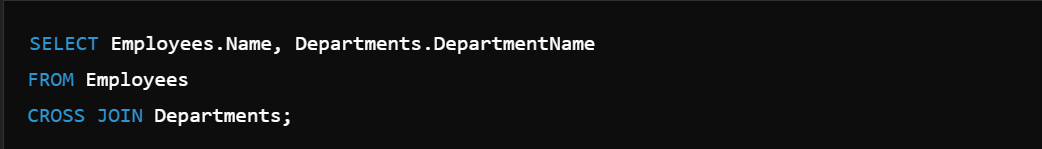
Cross Join returns the Cartesian product of two tables, meaning it combines every row from the first table with every row from the second table. It does not require any condition.

#### Key Features:

* Combines all rows from both tables
* Number of rows in the result = (Rows in Table 1) × (Rows in Table 2)
* Can produce large result sets if tables have many rows

#### Syntax:

**Example:**



This query returns all possible combinations of Employees and Departments, with every employee paired with every department.

## What is Union and Union All?

UNION and UNION ALL are used to combine the result sets of two or more SELECT statements.

**UNION**

Removes duplicate rows

Slower due to duplicate removal

Automatically sorts the result set

Syntax: UNION

**UNION ALL**

Includes duplicate rows

Faster as no duplicate removal

Does not sort the result set

Syntax: UNION ALL

### Syntax:



**Example (UNION):**

This query combines employee and manager names without duplicates.

### Example (UNION ALL):

This query combines employee and manager names, including duplicates.

## What is the difference between UNION and UNION ALL?

**Criteria Duplicates**

**Performance Sorting Usage**

**Syntax**

**UNION**

Removes duplicate rows

Slower (due to duplicate removal)

Automatically sorts the result set

Used when duplicate data is not required

SELECT column FROM table1 UNION SELECT column FROM table2;

**UNION ALL**

Includes all duplicate rows

Faster (no duplicate removal)

Does not sort the result set

Used when duplicate data needs to be preserved

SELECT column FROM table1 UNION ALL SELECT column FROM table2;

1. **What is Normalization?**

**EmployeeID**

101

102

**EmployeeName**

Vinay

Awadhesh

**DepartmentID**

1

1

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller related tables and defining relationships between them.

### Key Features:

* Reduces data redundancy
* Improves data consistency
* Simplifies data maintenance
* Increases data integrity

### Types of Normalization:

1. **1NF (First Normal Form)** – Eliminates duplicate columns and ensures each column contains atomic values.

2. **2NF (Second Normal Form)** – Ensures no partial dependency by making all non-key attributes fully dependent on the primary key.

3. **3NF (Third Normal Form)** – Removes transitive dependencies where non-key columns depend on other non-key columns.

4. **BCNF (Boyce-Codd Normal Form)** – Ensures that every determinant is a candidate key.

### Example: Unnormalized Table:

**EmployeeID**

101

102

**Employee Name**

Vinay Awadhesh

**Department**

IT IT

**DepartmentLocation**

Bangalore Bangalore

**Normalized Table (1NF & 2NF): Employee Table:**

**DepartmentID**

1

2

**DepartmentName**

IT

HR

**Department Table:**

Normalization improves the efficiency and consistency of the database.

**DepartmentID**

1

**DepartmentName**

IT

**Location**

Bangalore

## What is Denormalization?

Denormalization is the process of combining tables or adding redundant data into a database to improve read performance at the cost of data redundancy.

It is the opposite of Normalization, used when fast data retrieval is more important than maintaining data integrity.

### Key Features:

* Improves data retrieval speed
* Increases data redundancy
* Reduces the number of joins required
* Used in data warehouses and reporting systems

**Example:**

**Normalized Tables: Employee Table:**

**EmployeeID**

101

102

**EmployeeName**

Vinay

Awadhesh

**DepartmentID**

1

2

**Department Table:**

**Denormalized Table:**



In this example, Name will always take 10 bytes, while Address will use only the necessary space based on the actual data length.

In this example, the DepartmentName column is directly stored in the Employee table, making data retrieval faster but increasing redundancy.

**EmployeeID**

101

102

**EmployeeName**

Vinay

Awadhesh

**DepartmentName**

IT

HR

## What is the Difference Between CHAR and VARCHAR?

**Criteria Full Form Storage**

**Memory Usage**

**Performance Padding**

**Use Case**

**CHAR**

Character Fixed-length

Always uses the specified

length, even if fewer characters are stored

Faster for fixed-size data Pads extra spaces to match the

fixed length

When data length is consistent (like PIN codes or gender)

**VARCHAR**

Variable Character Variable-length

Uses only the space required for

the actual data plus one or two bytes for length

Slower for varying data sizes Does not pad spaces

When data length varies (like

names or addresses)

**Example:**

**24. What is the difference between SQL and MySQL?**

**MySQL**

Relational Database Management System (RDBMS) that uses SQL

Software

Used to store, manage, and retrieve data

Developed by Oracle Corporation

Specific to MySQL

Database system to store and manage data

**SQL**

Structured Query Language used to manage and manipulate databases

Language

Used to write queries to interact with databases

Standard language developed by ANSI

Universal (used by many databases like MySQL, Oracle, SQL Server)

Query language to communicate with databases

**Criteria Definition**

**Type Usage**

**Developer**

**Platform**

**Purpose**

**Example:**

* **SQL:**

****

* **MySQL:**

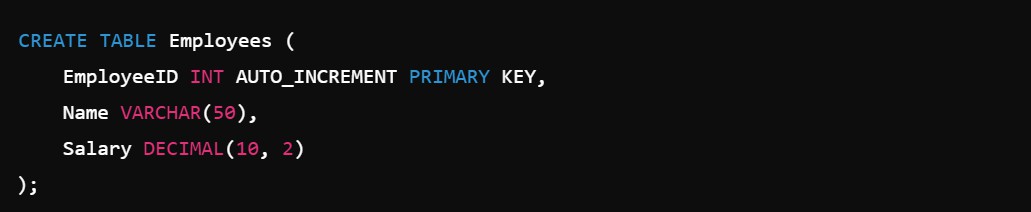
MySQL stores the Employees table and processes the SQL query to retrieve data.

## What is Auto Increment in SQL?

Auto Increment is a property in SQL that automatically generates a unique sequential number whenever a new row is inserted into a table. It is typically used to create unique identifiers like primary keys.

### Key Features:

* + Automatically generates unique numbers
  + Commonly used with Primary Key columns
  + Starts from a defined value (default is 1)
  + Automatically increments by 1 for each new row



**Syntax (MySQL):**

### Example (Insert Records):

**EmployeeID**

1

2

**Name**

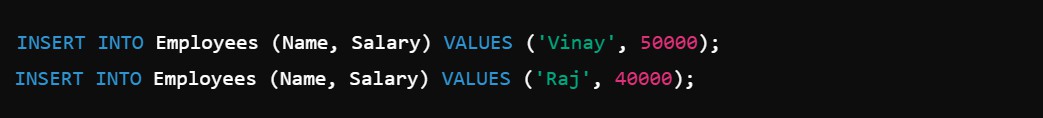
Vinay

Awadhesh

**Salary**

50000

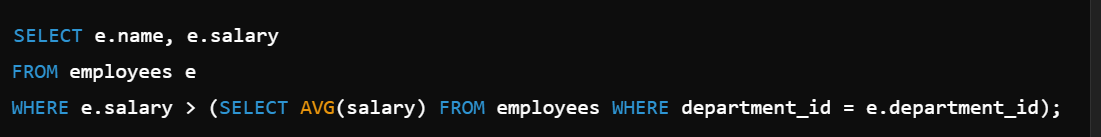
40000

****

**Output:**

The EmployeeID column automatically increments without user input.

**Intermediate Level**



**Explanation:** It selects employees whose salary is higher than the average salary of their own department.

## What is Subquery?

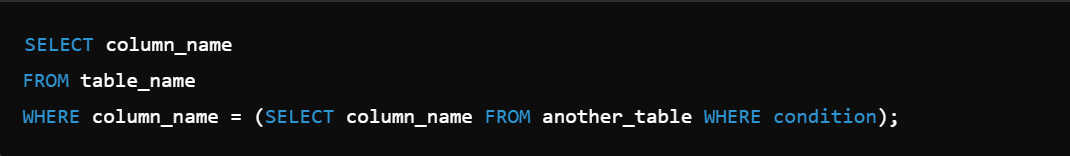
A Subquery is a query nested inside another query in SQL.

It is used to fetch data that will be used by the main query as a condition to filter or manipulate the result.

### Key Points:

* Also called Inner Query or Nested Query.
* Always executes before the main query.
* The result of the subquery is used by the Outer Query.
* Can be used with SELECT, INSERT, UPDATE, or DELETE statements.

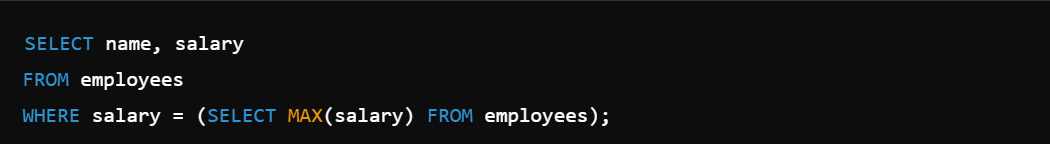
### Syntax:

****

**Types of Subqueries:**

1. **Single Row Subquery:** Returns only one value.

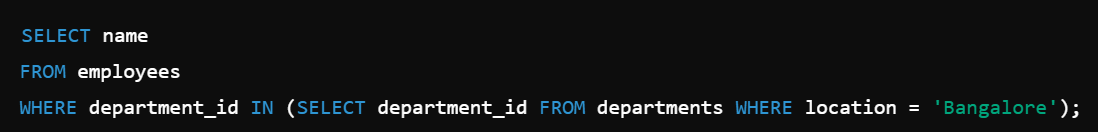
### Example:

****

**Explanation**: It selects the employee with the highest salary.

1. **Multiple Row Subquery:** Returns multiple rows of data.

### Example:

****

**Explanation**: It selects employees working in the Bangalore location.

1. **Correlated Subquery:** Uses each row of the outer query to execute the subquery.

**Example:**

## What is Nested Query?



A Nested Query is a query written inside another query to retrieve data based on the result of the inner query.

It helps break down complex queries into smaller, more manageable parts.

### Key Points:

* Also known as Inner Query or Subquery.
* The Inner Query executes first, and its result is passed to the Outer Query.
* Used in SELECT, INSERT, UPDATE, or DELETE statements.
* Can be used with WHERE, HAVING, and FROM clauses.

### Syntax:

**Example:**

Find employees who work in the 'Sales' department.



### Explanation:

* + The inner query finds the department\_id for the 'Sales' department.
  + The outer query selects employees based on that department\_id.

## What is Correlated Subquery?

A Correlated Subquery is a subquery that depends on the values from the outer query to execute.

It is executed repeatedly for each row of the outer query, making it slower compared to regular subqueries.

### Key Points:

* The Inner Query uses columns from the Outer Query.
* Executes row by row for each result of the outer query.
* Cannot be executed independently without the outer query.
* Used for row-by-row comparisons.

### Syntax:



**Example:**

Find employees whose salary is higher than the average salary of their department.

**Explanation:**

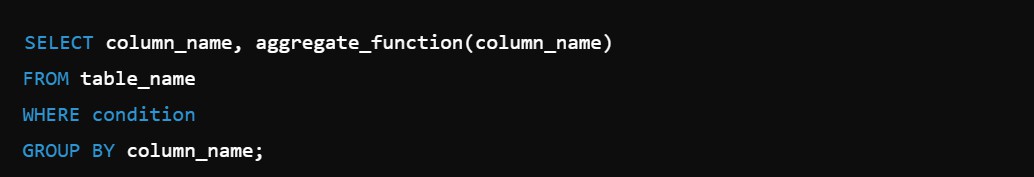
* The inner query calculates the average salary of each department.
* The outer query checks if the employee's salary is higher than their department's average salary.

## What is GROUP BY in SQL?

The GROUP BY clause in SQL is used to group rows that have the same values into summary rows. It is typically used with aggregate functions to perform calculations on each group of data. **Key Points:**

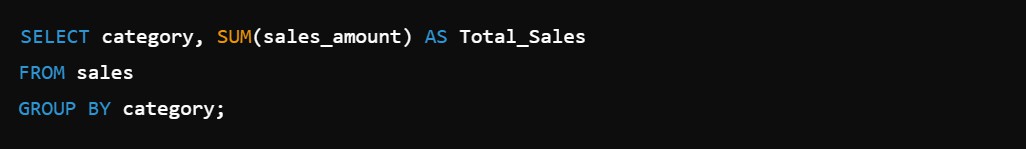
* + Groups data based on one or more columns.
  + It is used to summarize data.
  + Always used after the WHERE clause and before the ORDER BY clause.
  + Commonly used with aggregate functions like:
    - COUNT() – Counts the number of rows.
    - SUM() – Calculates the total sum.
    - AVG() – Calculates the average value.
    - MAX() – Returns the maximum value.
    - MIN() – Returns the minimum value.

### Syntax:

****

**Example:**

Find the total sales amount for each product category.



### Conclusion:

The GROUP BY clause is essential for data summarization and helps in analyzing data patterns efficiently.

## What is the Difference Between GROUP BY and ORDER BY in SQL?

Both GROUP BY and ORDER BY are SQL clauses used to organize query results, but they

serve different purposes.

**GROUP BY**

Used to group rows based on the same values in one or more columns.

Always works with aggregate functions like COUNT(), SUM(), AVG(), etc.

Syntax: Comes before ORDER BY. Groups the result into summary rows.

Example: Group sales by product category.

**ORDER BY**

Used to sort the result set in ascending or descending order.

Does not require aggregate functions.

Syntax: Always comes after GROUP BY. Sorts the entire result set.

Example: Sort sales by highest to lowest

amount.

### Example with GROUP BY:

Find the total sales for each product category.



### Example with ORDER BY:

Sort products by price in descending order.



### Conclusion:

* GROUP BY is used to group data and perform aggregate calculations.
* ORDER BY is used to sort the final result.



**31. What is the Use of LIMIT in SQL?**

LIMIT is used to restrict the number of rows returned by a query.

**Syntax:**

### Example:

Fetch top 3 highest salaries:



### Conclusion:

* LIMIT helps to fetch limited data and is commonly used for Top N records or Pagination.

## How to Find the Second Highest Salary in SQL?

### Using Subquery:

****

**Explanation:**

 The inner query gets the highest salary.

 The outer query finds the highest salary below the top salary.

## How to find Duplicate Records in a table?

You can find Duplicate Records using the GROUP BY clause with the HAVING keyword.

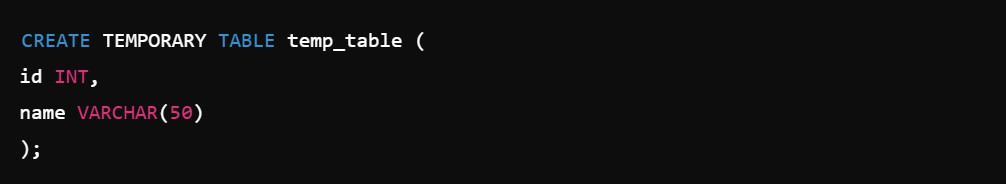
### Syntax:

****

**Explanation:**

* GROUP BY groups the same names.
* HAVING COUNT(\*) > 1 filters only duplicate names.

## What is CTE (Common Table Expression)?



CTE (Common Table Expression) is a temporary result set that is defined within the execution of a single SQL statement.

### Syntax:

****

**Example:**

Find employees who work in the IT department and have a salary greater than 50000.



### Explanation:

* The CTE IT\_Employees selects all employees from the IT department.
* The SELECT query fetches employees with salary above 50000.

## What is Temporary Table in SQL?

A Temporary Table in SQL is used to store temporary data during the session.

### Key Points:

* + Automatically deleted when the session ends.
  + Used to store intermediate results.
  + Prefixed with # in SQL Server or TEMP in MySQL.

### Syntax:

**MySQL:**



**SQL Server:**



### Example:

Create a temporary table to store employees with salary above 50000.



### Explanation:

* The table holds filtered data temporarily.
* It is automatically deleted after the session ends.

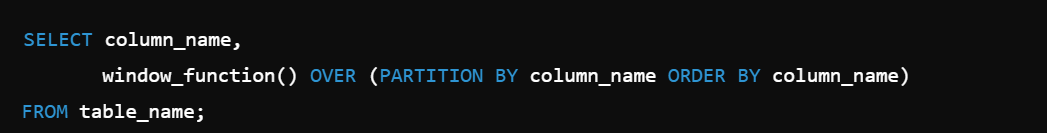
## What is Window Function in SQL?

A Window Function performs calculations across a set of table rows related to the current row without collapsing the result into a single value.

### Key Points:

* + Works with OVER() clause.
  + Does not group rows like aggregate functions.
  + Commonly used for ranking, running totals, and moving averages.

**Syntax:**

****

### Types of Window Functions:

**1. ROW\_NUMBER()**

Assigns a unique sequential number to each row in a partition.

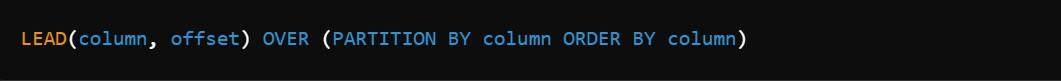
### Syntax:



**2. RANK()**

Assigns a rank to each row with the same values having the same rank, but skips ranks for duplicate values.

**Syntax:**



1. **DENSE\_RANK()**

Similar to RANK(), but does not skip ranks for duplicate values.

### Syntax:

****

1. **NTILE(n)**

Divides the result set into n equal parts and assigns a group number to each row.

### Syntax:

1. **SUM()**

Calculates the cumulative total of a column within a partition.

### Syntax:

****

1. **AVG()**

Calculates the average value of a column within a partition.

### Syntax:

****

1. **MAX() & MIN()**

Returns the maximum or minimum value in a partition.

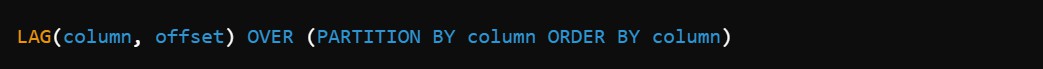
### Syntax:

****

1. **LEAD()**

Returns the next row's value in the result set.

**Syntax:**



**9. LAG()**

Returns the previous row's value in the result set.

**Syntax:**

**Summary Table:**

**Function**

ROW\_NUMBER

RANK

DENSE\_RANK

NTILE

SUM

AVG

LEAD

LAG

**Purpose**

Unique Rank

Rank with Skips

Rank without Skips

Divide Rows

Running Total

Running Average

Next Row Value

Previous Row Value

**Duplicates**

No

Yes

Yes

No

No

No

No

No

**Skips**

**Numbers**

No

Yes

No

No

No

No

No

No

**Example Usage**

Employee Ranking

Competition Rank

Class Rank

Quartiles

Salary Analysis

Salary Average

Price Trends

Sales Trends

**Conclusion:**

Window Functions help in performing complex calculations across result sets without grouping them into a single row, making them essential for ranking, running totals, and trend analysis.

## What is the Difference Between ROW\_NUMBER(), RANK(), and DENSE\_RANK()?



These three Window Functions are used to assign numbers to rows based on their order.

**Key Differences:**

Function

ROW\_NUMBE R()

RANK()

DENSE\_RAN K()

Purpose

Assigns unique sequential numbers to each row

Assigns rank to each row with the same values having the same rank

Assigns rank without skipping numbers for duplicate values

Duplicates

No

Yes

Yes

Skips

Numbers No

Yes

No

Example

1, 2, 3, 4

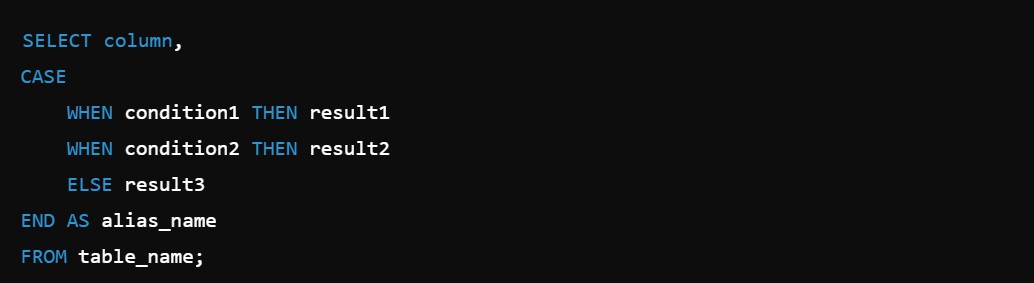
1, 2, 2, 4

1, 2, 2, 3

## What is CASE Statement in SQL?

The CASE Statement is used to apply conditional logic in SQL queries, similar to IF-ELSE statements.

**Syntax:**

****

**Example:**



**39. What is COALESCE in SQL?**

COALESCE returns the first non-null value from a list of expressions.

**Syntax:**

**Example:**

****

**Key Points:**

* Returns the first non-null value.
* Used to handle NULL values.
* Can accept multiple expressions.

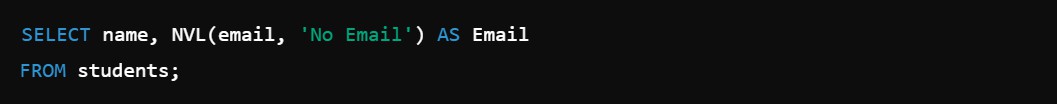
## What is NVL Function in SQL?

NVL function replaces NULL values with a specified value.

**Syntax:**

****

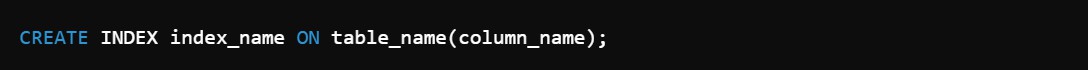
**Example:**

****

## What is Indexing in SQL?

Indexing improves the speed of data retrieval from a table by creating a lookup structure.

**Syntax:**

****

**Example:**

****

**Key Points:**

* Speeds up SELECT queries.
* Automatically maintained by the database.
* Can be created on one or more columns.
* Slows down INSERT, UPDATE, DELETE operations.



**42. What is Clustered Index in SQL?**

A Clustered Index sorts and stores the data physically in the table based on the indexed column.

**Syntax:**

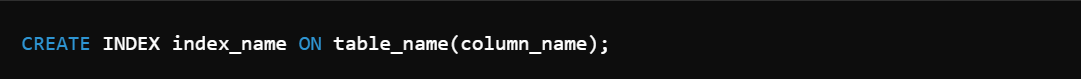
### Example:

**Key Points:**

* + Only one clustered index is allowed per table.
  + Faster for data retrieval.
  + Automatically created on Primary Key by default.
  + Rearranges table rows physically.

## What is Non-Clustered Index in SQL?

A Non-Clustered Index creates a separate structure from the table data, storing pointers to the actual rows.

**Syntax:**

**Example:**

****

**Key Points:**

* Multiple Non-Clustered Indexes can be created on a table.
* Improves search performance.
* Does not affect the physical order of data.
* Stores pointers to the actual data.

## Difference between Clustered and Non-Clustered Index

**Clustered Index**

Stores data physically sorted.

Only one per table.

Faster for data retrieval.

Automatically created on Primary Key.

Affects physical order of table.

**Non-Clustered Index**

Stores pointers to data.

Multiple indexes allowed.

Slower than Clustered Index.

Manually created on any column.

Does not affect physical order.



**45. What is View in SQL?**

A View is a virtual table based on the result of a SQL query.

**Syntax:**

### Example:

****

**Key Points:**

* + Does not store data physically.
  + Simplifies complex queries.
  + Provides data security.
  + Can be used like a table in SELECT queries.

## Difference between View and Table

**View**

Virtual table.

Does not store data physically.

Based on SQL queries.

Provides data security by restricting access to certain columns.

Automatically updates when base table data changes.

**Table**

Physical table.

Stores data physically.

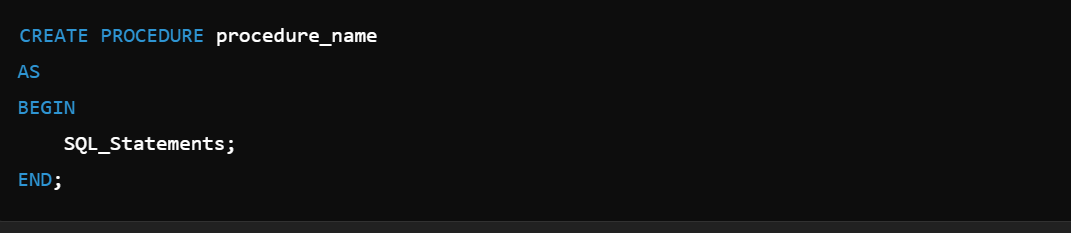
Contains raw data.

No data restriction unless applied.

Needs manual updates.

1. **What is Stored Procedure ?**

A Stored Procedure is a group of predefined SQL statements stored in the database that can be executed multiple times.

**Syntax:**

**Example:**

****

**Key Points:**

* Improves code reusability.
* Increases performance.
* Supports input and output parameters.
* Provides security by hiding SQL code.

## What is the difference between Function and Stored Procedure?

**Function**

Returns a single value or table.

Can be used in SELECT statements.

Allows only input parameters.

Cannot modify database state.

Always returns a value.

**Stored Procedure**

May or may not return a value.

Cannot be used in SELECT statements. Allows input and output parameters.

Can modify database state (INSERT, UPDATE, DELETE).

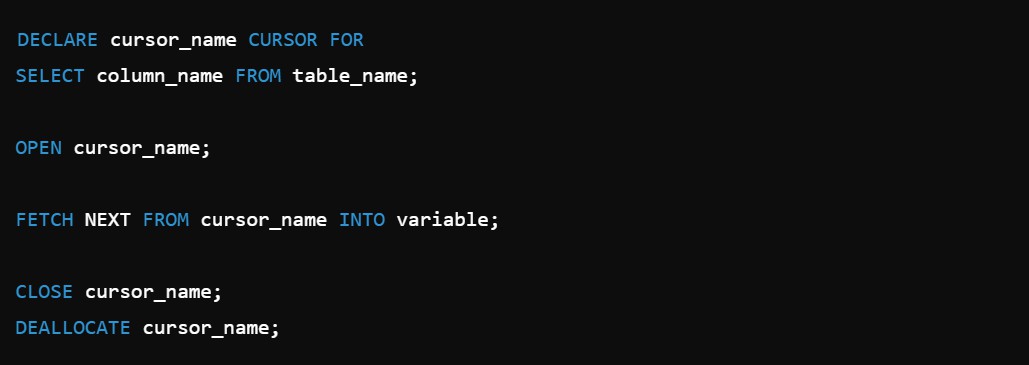
Does not always return a value.



**49. What is Trigger in SQL?**

A Trigger is an automatic action executed when a specified event occurs in a table.

**Syntax:**



### Example:

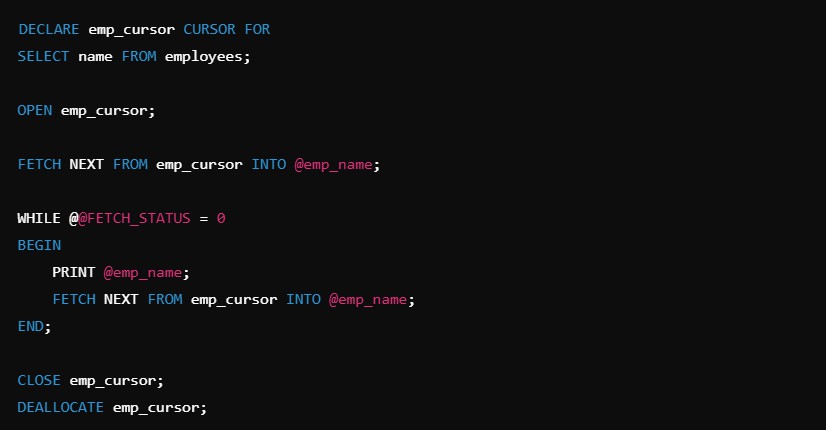
**Key Points:**

* + Automatically executes on INSERT, UPDATE, or DELETE.
  + Used for data validation and logging.
  + Cannot be called manually.
  + Improves data integrity.

## What is Cursor in SQL?

A Cursor is a database object used to retrieve, manipulate, and navigate row-by-row through the result set.

### Syntax:



**Example:**

**Key Points:**

* + Used to process row-by-row results.
  + Slower than set-based operations.
  + Helps in complex data manipulation.
  + Not recommended for large datasets.

**Advanced Level**

## What is the ACID Property in SQL?

The ACID properties in SQL define the key principles to ensure that database transactions are processed reliably without affecting data integrity.

**ACID Stands for:**

**Property**

A - Atomicity

C - Consistency

I - Isolation

D - Durability

**Description**

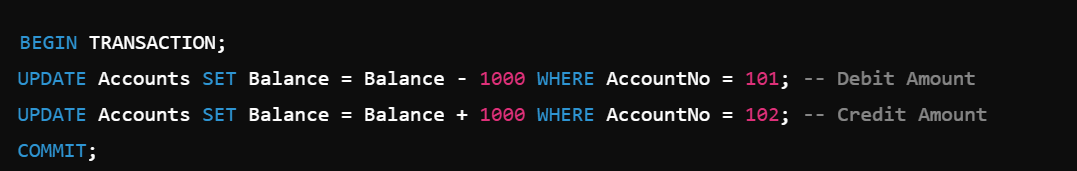
Transaction should be all or nothing. If one part of the transaction fails, the entire transaction fails, and the database remains unchanged.

The database must be in a consistent state before and after the transaction. It ensures that data remains correct and valid.

Transactions should be executed independently, without interfering with each other.

Once a transaction is committed, the changes must be permanent in the database, even if the system crashes.

**Example:**

****

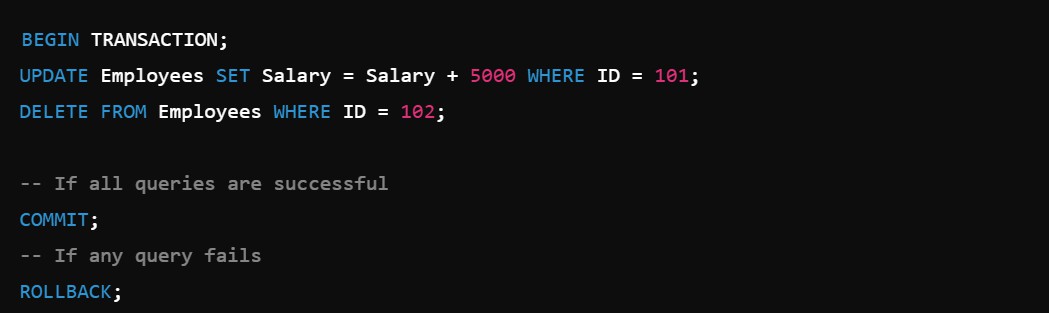
**Explanation:**

1. **Atomicity:** If one of the two queries fails, both updates will be rolled back. 2. **Consistency:** The total amount in both accounts will remain the same.

3. **Isolation:** If another transaction is trying to access the same account, it will wait until this transaction completes.

4. **Durability:** After COMMIT, changes will be saved permanently even in case of a power failure.

## What is a Transaction in SQL?



A Transaction in SQL is a group of SQL operations that are executed as a single unit to perform a specific task on the database.

It follows ACID Properties to maintain data integrity.

**Key ACID Properties:**

**Property**

Atomicity

Consistency

Isolation

Durability

**Description**

All operations must succeed or none will happen.

The database must remain in a valid state before and after the transaction.

Transactions execute independently without affecting each other.

Once committed, changes are permanent even after system failure.

**Command**

BEGIN TRANSACTION

COMMIT

ROLLBACK

SAVEPOINT

**Description**

Starts a new transaction.

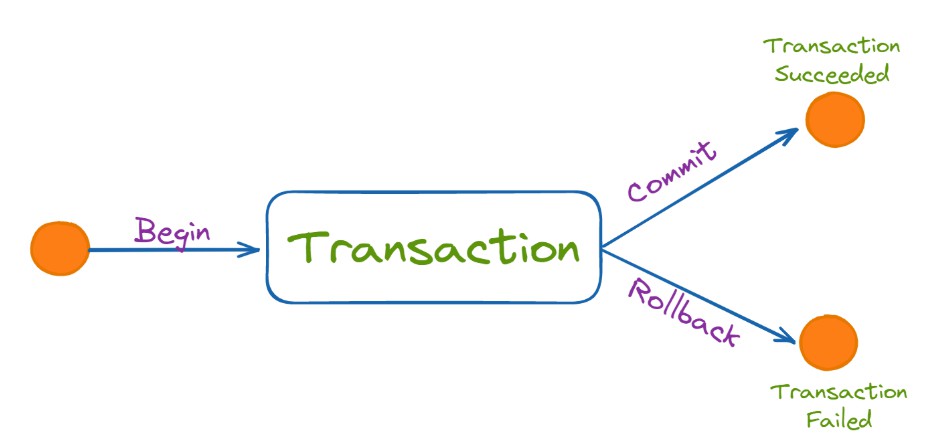
Saves the changes permanently.

Undo changes if any error occurs.

Sets a point to roll back to partially.

**Transaction Commands:**

**Example:**



**Created by: Vinay Kumar Panika**

**Transactions in SQL**

## What is the difference between COMMIT and ROLLBACK?

**COMMIT**

Saves the changes made by the transaction permanently into the database.

Once executed, changes cannot be undone.

Syntax: COMMIT;

Used when all operations are successful.

Improves data durability.

**ROLLBACK**

Undo all changes made by the transaction.

Restores the database to its previous state.

Syntax: ROLLBACK;

Used when any error occurs during the transaction.

Helps to maintain data consistency.

**Example:**

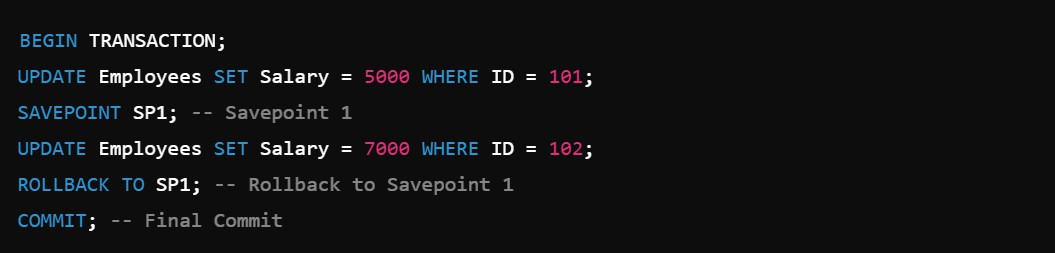
## What is Savepoint in SQL?

Savepoint in SQL is used to temporarily save a transaction at a specific point, allowing you to rollback only part of the transaction without affecting the entire transaction.

### Key Points:

* + Allows setting multiple points in a transaction.
  + Helps in partial rollback.
  + Improves error handling.
  + Used with ROLLBACK.

### Syntax:

****

**Explanation:**

* The first update will be saved.
* The second update will be rolled back.
* Remaining changes will be committed.

## What is the difference between IN and EXISTS?

**IN**

Compares values from the main query with a list of values.

Works with static values or subqueries.

**Slower with large datasets.**

Returns all matching rows.

**EXISTS**

Checks if subquery returns any rows.

Only works with subqueries.

**Faster for large datasets.**

Stops checking after finding the first match.

**56. What is the difference between DELETE and TRUNCATE?**

**57. What is Index Fragmentation?**

**DELETE**

Removes specific rows based on a condition using the WHERE clause.

Can be rolled back using ROLLBACK if inside a transaction.

Slower because it logs each row deletion.

Maintains table structure and identity column values.

**TRUNCATE**

Removes all rows from the table without any condition.

Cannot be rolled back once executed.

Faster because it does not log individual row deletions.

Resets identity column values to the initial seed.

Index Fragmentation occurs when the logical order of index pages in the database does not match the physical order of data on disk, making data retrieval slower.

### Types of Index Fragmentation:

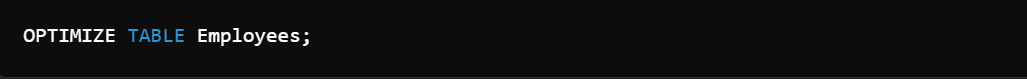
1. **Internal Fragmentation –** Unused space inside index pages due to data deletion or updates.
2. **External Fragmentation –** Index pages are stored in non-sequential order, causing slower data access.

### How to Check Index Fragmentation in MySQL:

****

**How to Fix Index Fragmentation:**

 Use OPTIMIZE TABLE to reorganize index pages.



### Conclusion:

Index fragmentation slows down query performance and should be fixed regularly to maintain database efficiency.

**UNION**

Combines result sets vertically (rows) from two or more tables.

Removes duplicates by default (UNION), or includes duplicates with UNION ALL.

Tables should have the same number of columns and data types.

Used when tables have similar data.

**JOIN**

Combines result sets horizontally (columns) based on common columns.

Does not remove duplicates.

Tables can have different numbers of columns.

Used when tables have related data through a common column.

**58. What is the difference between RANK() and DENSE\_RANK()?**

## How to fetch common records from two tables?

**RANK()**

Assigns a unique rank to each row, but skips the next rank if there are duplicate values.

Gaps are created in ranking sequence.

Slower in performance compared to DENSE\_RANK().

**DENSE\_RANK()**

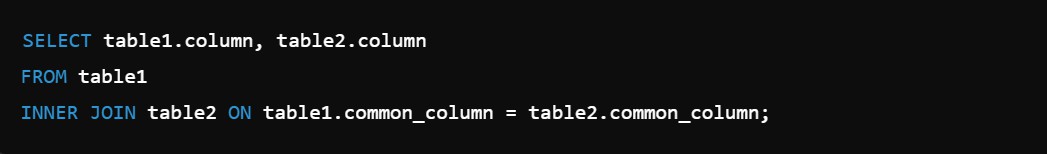
Assigns a unique rank to each row without skipping ranks if there are duplicate values.

No gaps in ranking sequence.

Faster than RANK() because it doesn't skip ranks.

To fetch common records from two tables, you can use the INNER JOIN clause in SQL.

**Syntax:**

****

## What is the difference between UNION and JOIN?

1. **What is Pivot Table in SQL?**



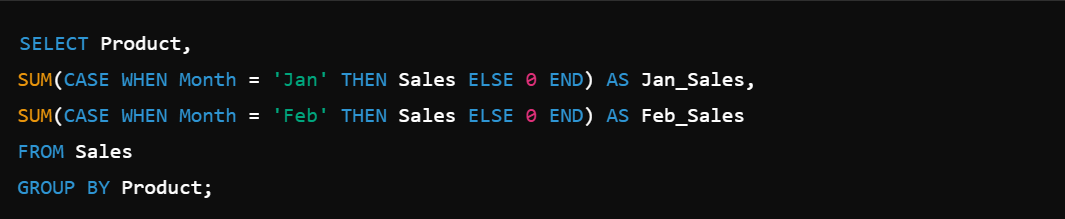
A Pivot Table in SQL is used to transform row data into column data to provide a summary report of the dataset.

It is commonly used to perform data aggregation and present data in a more readable format.

### Key Points:

* Converts rows into columns.
* Used to generate summary reports.
* Helps in data analysis.
* Commonly used with aggregate functions like SUM(), AVG(), COUNT(), etc.

### Example: Pivot Table Query

****

**Conclusion:**

Pivot Tables help to summarize large datasets and present them in a structured format.

## What is Case Sensitivity in SQL?

Case Sensitivity in SQL refers to whether the database treats uppercase and lowercase letters as different or same when performing queries.

### Key Points:

* + **Column Names:** Most databases are not case-sensitive (MySQL, SQL Server).
  + **Table Names:** Case sensitivity depends on the database and operating system.
  + **String Values:** By default, MySQL is case-insensitive for string comparisons.

### Example in MySQL:

**Output:**

**How to Make MySQL Case-Sensitive:**

**EmpID**

101

102

**Name**

Vinay

VINAY

****

This query will only return exact case-sensitive matches.

### Conclusion:

* Case Insensitivity is default in MySQL for string comparisons.
* Use the BINARY keyword to perform case-sensitive searches.

## How to find the Nth Highest Salary?

To find the Nth Highest Salary in SQL, you can use the LIMIT with OFFSET method or Subquery with ORDER BY.

**Method 1: Using LIMIT with OFFSET (MySQL)**

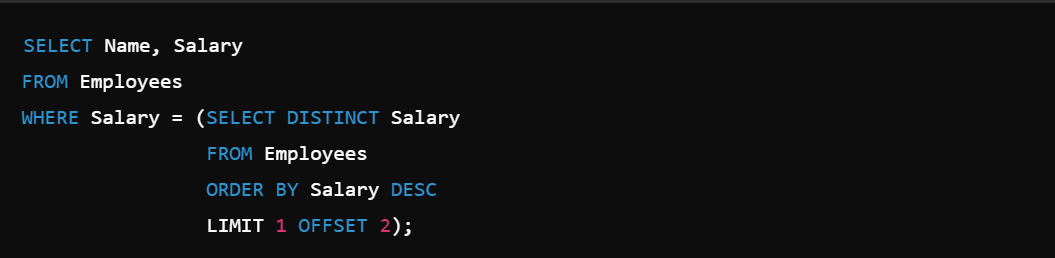
****

**Example: Example (3rd Highest Salary):**

****

**Explanation:**

* **ORDER BY Salary DESC** → **Sorts the salaries in descending order.**
* **OFFSET 2** → **Skips the top 2 salaries.**
* **LIMIT 1** → **Selects the next salary as the 3rd highest.**



**Method 2: Using Subquery with LIMIT**

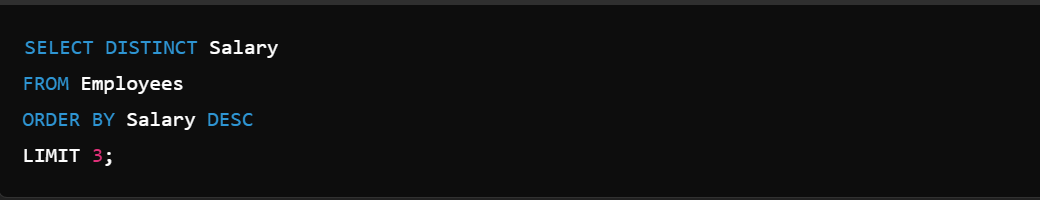
**Conclusion:**

* + Use LIMIT with OFFSET for faster results in MySQL.
  + This method is commonly asked in interviews.
  + Always use DISTINCT to remove duplicate salaries.

## How to get First 3 Maximum Salaries?

To fetch the First 3 Maximum Salaries from a table, you can use the DISTINCT, ORDER BY, and LIMIT clauses.

### Method : Using LIMIT (MySQL)

****

**Explanation:**

* **DISTINCT** → Removes duplicate salaries.
* **ORDER BY Salary DESC** → Sorts salaries in descending order.
* **LIMIT 3** → Fetches the top 3 salaries.

**65. What is the difference between Drop, Delete, and Truncate?**

### Conclusion:

**Comman d**

DROP

DELETE

TRUNCAT E

**Purpose**

Deletes the entire table with its structure

Removes specific rows based on condition

Removes all rows from the table

**Can Rollback**

No

Yes (if within Transaction

)

No

**Speed**

Fastest

Slow

Fast

**Structure Affected**

Yes (Removes table structure)

No

No (Keeps structure)

**Condition**

**-Based**

No

Yes (With WHERE

Clause)

No

* Use DELETE for removing specific rows with conditions.
* Use TRUNCATE for removing all rows quickly without rollback.
* Use DROP to delete both data and table structure completely.

## 66. How to calculate Age from Date of Birth in SQL?

You can calculate the Age from the Date of Birth using the DATEDIFF() or YEAR() functions depending on the database.

### Method 1: Using DATEDIFF() (MySQL)

****

**Explanation:**

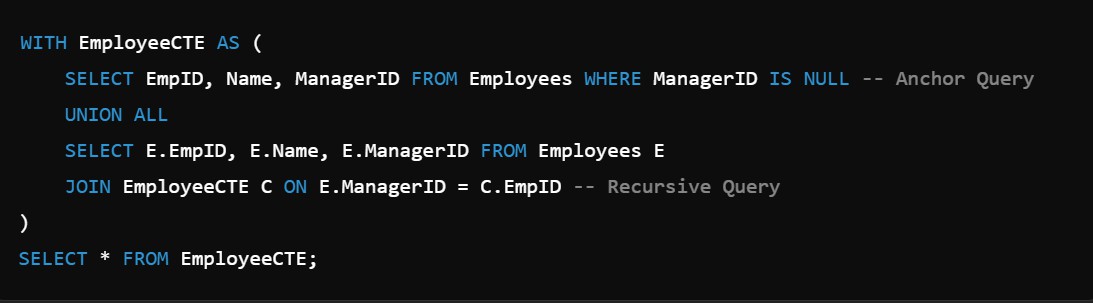
 **CURDATE()** → Returns the current date.

 **DATEDIFF()** → Calculates the difference between the current date and date of birth in days.

 **FLOOR()** → Converts the result into whole years.



**Method 2: Using YEAR() (MySQL)**



### Explanation:

This method calculates the difference between the current year and the birth year.

### Conclusion:

 Use DATEDIFF() for accurate age calculation.

 Use YEAR() for simple year-based age calculation.

## 67. What is Recursive Query in SQL?

A Recursive Query in SQL is a query that refers to itself to perform repetitive operations until a specific condition is met. It is commonly used to process hierarchical data such as employee-manager relationships or organizational structures.

### Key Points:

* Used to handle hierarchical data.
* Implemented using Common Table Expressions (CTE).
* The recursion continues until the termination condition is satisfied.

**Example: Employee Table:**

**EmpID**

101

102

103

**Name**

Vinay

Awadhesh

Nevendra

**ManagerID**

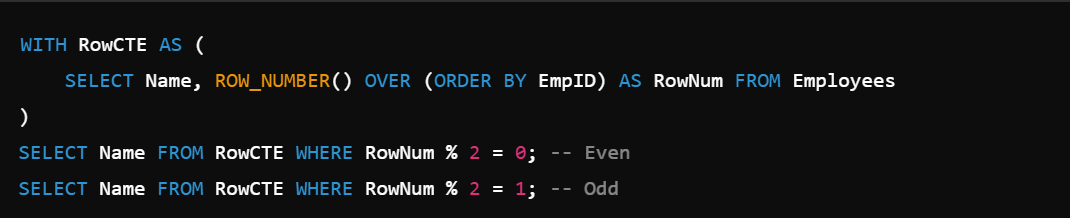
NULL

101

102

**Recursive Query:**

**68. What is the difference between Temporary Table and CTE?**



**Conclusion:**

**Temporary Table**

Stores data physically in temporary memory.

Needs to be explicitly created and dropped.

Can be used multiple times within a session.

Supports Indexing and DDL operations.

Slower for small datasets.

**CTE (Common Table Expression)**

Stores data logically without physical storage.

Automatically disappears after query execution.

Can be used only once in the same query.

Does not support Indexing or DDL operations.

Faster for small datasets.

* Use Temporary Tables when data needs to be reused multiple times.
* Use CTE for short-term data manipulation and improved readability.

1. **How to find Odd and Even records in SQL?** You can find Odd and Even records in SQL using the MOD() or ROW\_NUMBER() functions.

### Method 1: Using MOD() Function (MySQL)

**Explanation:**

* + MOD(EmpID, 2) → Returns the remainder when EmpID is divided by 2.
  + If the remainder is 0, the record is Even.
  + If the remainder is 1, the record is Odd.

### Method 2: Using ROW\_NUMBER() (SQL Server, PostgreSQL)

**Conclusion:**

Use MOD() for databases like MySQL.

Use ROW\_NUMBER() for databases that support Window Functions.

## What is JSON in SQL?

JSON (JavaScript Object Notation) in SQL is used to store, retrieve, and manipulate data in a structured, text-based format within relational databases.

### Key Points:

* + Stores data in key-value pairs.
  + Lightweight and easy to read.
  + Commonly used for semi-structured data.
  + Supported in MySQL, SQL Server, and PostgreSQL.

### MySQL Example:

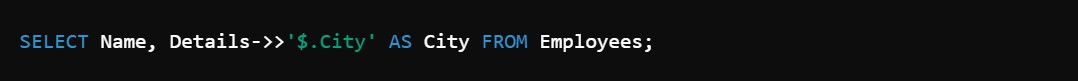
**Create Table with JSON Column:**

****

**Insert JSON Data:**

****

**Retrieve JSON Data:**

****

**Conclusion:**

JSON helps to handle semi-structured data within relational databases without the need for separate NoSQL databases.

## What is XML in SQL?

XML (Extensible Markup Language) in SQL is used to store, retrieve, and manipulate structured data in a text-based format within relational databases.

### Key Points:

* + Stores data in hierarchical format using tags.
  + Commonly used for data exchange between applications.
  + Supported in SQL Server, MySQL, and Oracle.
  + Helps to store semi-structured data.



**Conclusion:**

Use COALESCE() or IFNULL() to replace NULL values and ensure data consistency in queries.

**Created by: Vinay Kumar Panika**

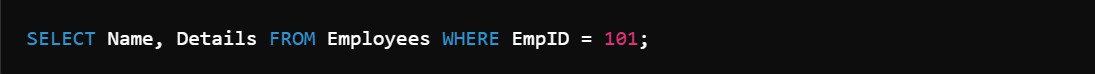
**Insert XML Data:**



**MySQL Example:**

**Create Table with XML Data:**

**Retrieve XML Data:**

****

**Conclusion:**

XML is used to store and transfer hierarchical data in relational databases, making it easier to exchange data between applications.

## How to handle NULL values in SQL?

NULL represents missing or unknown data in SQL.

### Methods to Handle NULL Values:

1. **IS NULL –** To check if a column contains NULL.



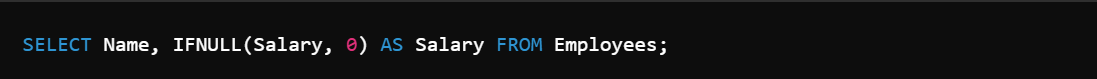
1. **IS NOT NULL –** To check if a column does not contain NULL.



1. **COALESCE() –** Replaces NULL with a default value.



1. **IFNULL() (MySQL) –** Replaces NULL with a specified value.



1. **NULLIF() –** Returns NULL if two expressions are equal.

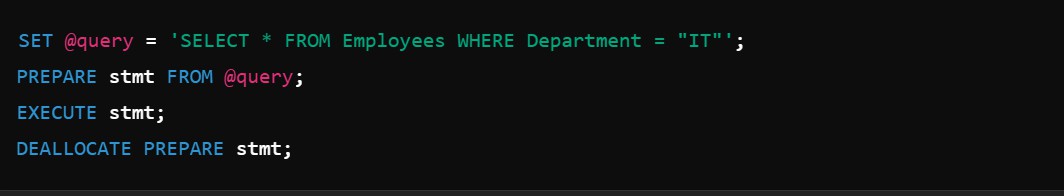
## What is Dynamic SQL?

Dynamic SQL is a method of constructing and executing SQL statements at runtime instead of writing static queries.

### Key Points:

* + Allows flexible query creation based on user input or conditions.
  + Used for complex queries with varying conditions.
  + Helps in Parameterized Queries and stored procedures.
  + Increases security risks if not handled properly.

### Example (MySQL):

****

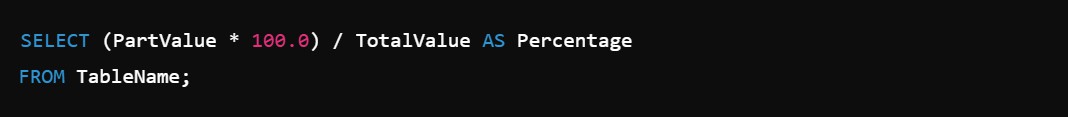
**Conclusion:**

Dynamic SQL provides flexibility in query execution but should always be used with parameterized queries to prevent SQL injection attacks.

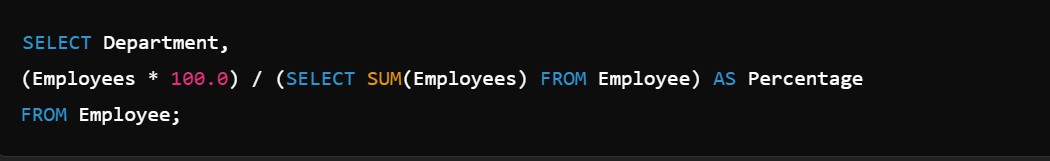
## How to calculate Percentage in SQL?

You can calculate Percentage in SQL using arithmetic expressions and aggregate functions.

### Syntax:

****

**Example:** Query to Calculate Percentage of Employees in Each Department:



## How to find the Employees who earn more than their Manager?

To find employees who earn more than their manager, you need to join the Employee table with itself using Self Join.

## Example Table:

**Employee Table**

**EmpID**

101

102

103

104

**Name**

Vinay Nevendra Awadhesh Rituraj

**Salary**

50000

40000

60000

45000

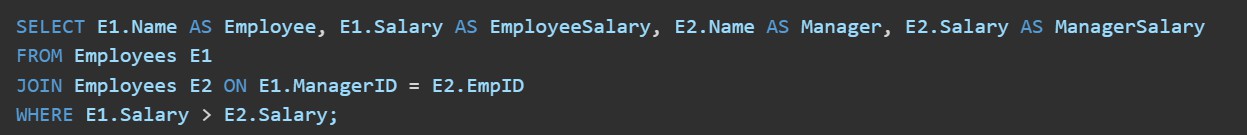
**ManagerID**

NULL 101

101

102

**Query:**



**Output:**

**Employee**

Awadhesh

**EmployeeSalary**

60000

**Manager**

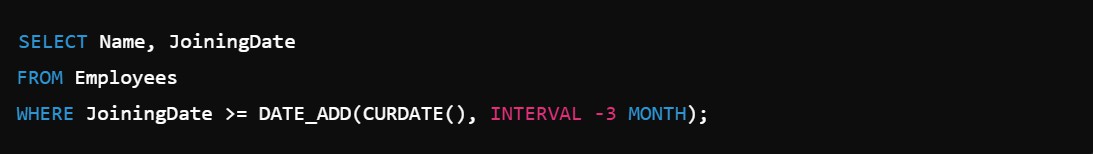
Vinay

**ManagerSalary**

50000

**Conclusion:** Use Self Join with a condition to compare employee salaries against their managers' salaries. This query helps in hierarchical data analysis.

**Real-Time Scenarios**



## How to find Duplicate Emails in the Employee Table?

To find duplicate emails in SQL, you can use the GROUP BY clause with the HAVING condition.

## Syntax:

**Explanation:**

* GROUP BY groups the records based on the Email column.
* COUNT(Email) counts how many times each email appears.
* HAVING COUNT(Email) > 1 filters only those emails that have more than one occurrence.

## How to get the Highest Salary in each Department?

You can find the Highest Salary in Each Department using the GROUP BY clause with the MAX() aggregate function.

## Syntax:

1. **How to find Employees joined in the last 3 months?** To find employees who joined in the last 3 months, you can use the DATE\_ADD() or DATEDIFF() function along with the WHERE clause.

**Method 1: Using DATEDIFF() (MySQL)**

****

**Method 2: Using DATE\_ADD() (MySQL)**

1. **How to Display the First 5 Records in SQL?**

You can display the First 5 Records using the LIMIT or TOP clause depending on the database.

## Method 1: Using LIMIT (MySQL)

****

**Method 2: Using TOP (SQL Server)**



1. **How to find the Number of Employees in each Department?** You can find the Number of Employees in each department using the COUNT() function with the GROUP BY clause.

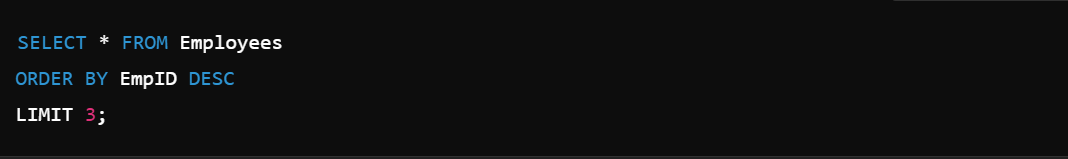
## Syntax:

****

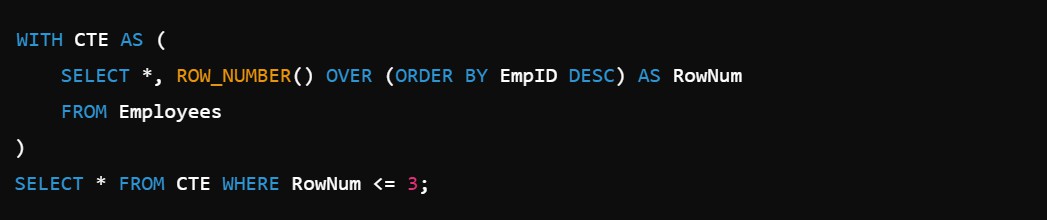
1. **How to find the Last 3 Records in SQL?**

To fetch the Last 3 Records in SQL, you can use the ORDER BY clause along with LIMIT.

**Method 1: Using ORDER BY with LIMIT (MySQL)**

****

**Method 2: Using ROW\_NUMBER() (SQL Server, PostgreSQL)**

****



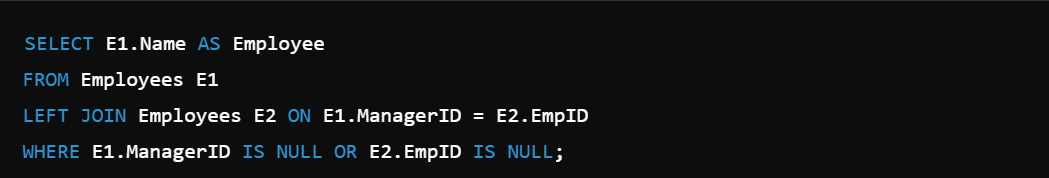
**82. How to find Employees without Managers?**

To find employees without managers, you need to filter records where the ManagerID column is NULL or does not exist in the table.

**Method 1: Using IS NULL (MySQL, SQL Server, Oracle)**



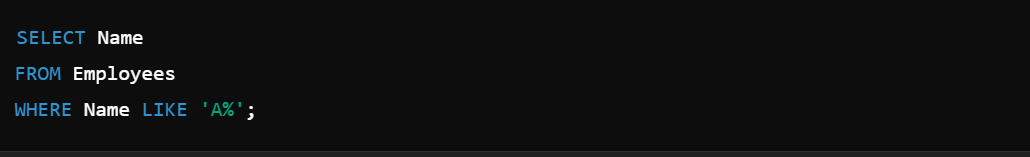
**Method 2: Using LEFT JOIN (MySQL, SQL Server, PostgreSQL)**

****

1. **How to find the First Name starting with 'A'?**

You can find employee names starting with the letter 'A' using the LIKE operator.

## Syntax:

****

**Explanation:**

* **LIKE 'A%'** → Finds names that start with 'A'.
* **%** → Represents any number of characters after 'A'.

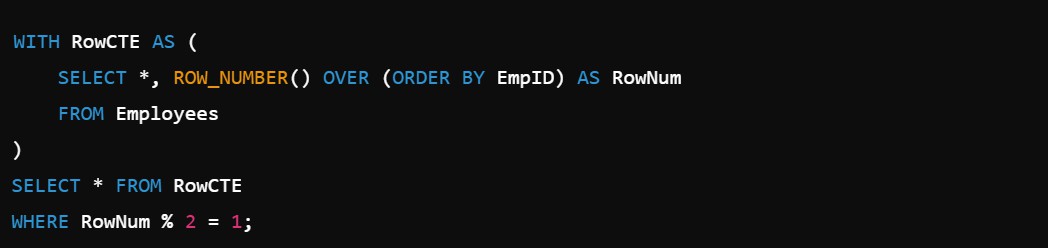
## How to fetch Alternate Rows from a table?

You can fetch Alternate Rows using the MOD() or ROW\_NUMBER() functions based on row position.

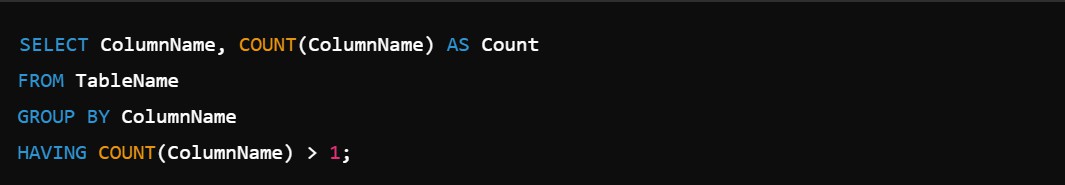
**Method 1: Using MOD() (MySQL) Fetch Even Rows:**



**Fetch Odd Rows:**



**Method 2: Using ROW\_NUMBER() (SQL Server, PostgreSQL) Fetch Odd Rows:**

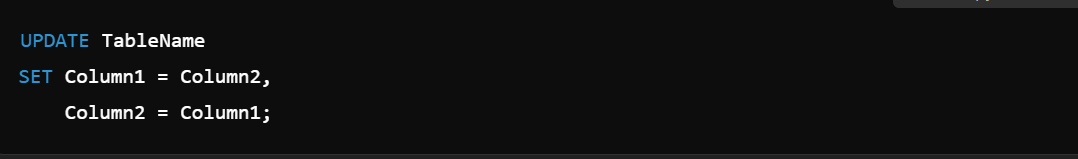


**Fetch Even Rows:**

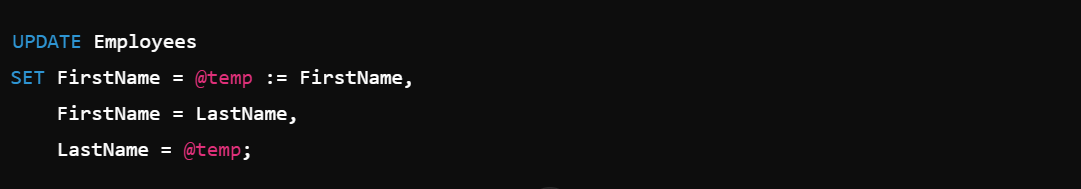
1. **How to swap two columns in SQL?**

You can swap the values of two columns using the UPDATE statement with the TEMPORARY variable technique.

## Syntax:

****

**Correct Way Using Temporary Variable:**

****

1. **How to display the Duplicate Records with their Count?** To display duplicate records along with their occurrence count, use the GROUP BY clause with the HAVING clause.

## Syntax:

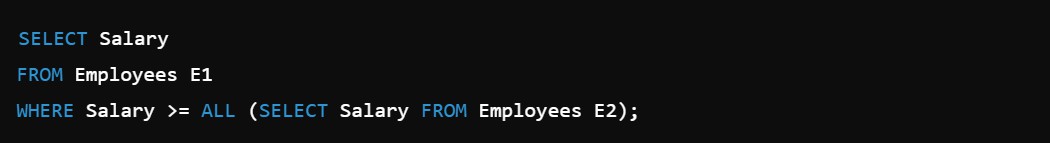
1. **How to find the Highest Salary without using MAX()?** You can find the Highest Salary without using the MAX() function by using the ORDER BY clause with the LIMIT or TOP keyword.



**Method 1: Using ORDER BY with LIMIT (MySQL)**

****

**Method 2: Using Subquery (MySQL, SQL Server)**

****

**Method 3: Using NOT IN (MySQL, SQL Server)**

****

**Conclusion:**

* Use ORDER BY with LIMIT for simple queries.
* Use Subqueries or ALL for advanced scenarios.
* This method works in all RDBMS.

## How to fetch common records from two tables without JOIN?

You can fetch common records from two tables without using JOIN by using the IN or INTERSECT operators.

**Method 1: Using IN (MySQL, SQL Server)**

****

**Method 2: Using INTERSECT (SQL Server, PostgreSQL)**



**89. How to delete Duplicate Records from a table?**

You can delete duplicate records using CTE, ROW\_NUMBER(), or GROUP BY methods.

**Method 1: Using ROW\_NUMBER() (SQL Server, PostgreSQL)**

**Method 2: Using GROUP BY with MIN() (MySQL)**

****

**Method 3: Using DISTINCT INTO Temporary Table (MySQL)**



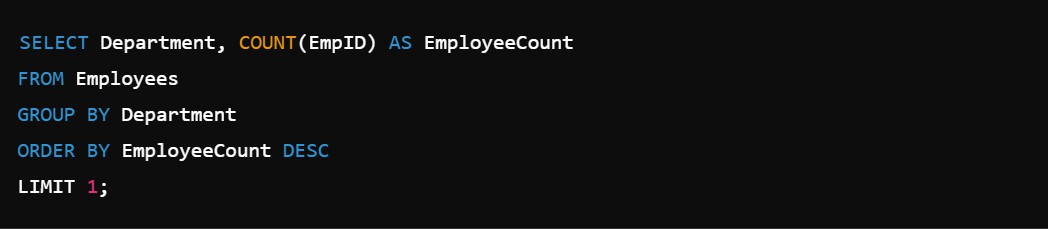
**Conclusion:**

* Use ROW\_NUMBER() for advanced databases.
* Use GROUP BY for simple queries.
* Always backup data before deleting duplicates.

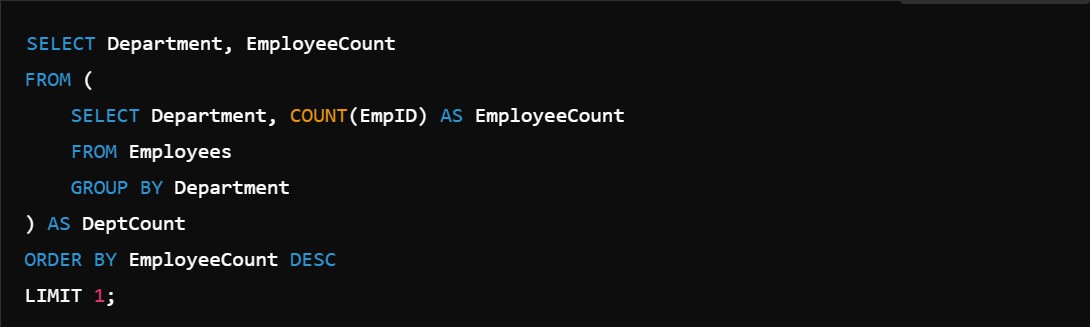
## How to find the Department with the highest Employee Count?

You can find the Department with the Highest Employee Count using the GROUP BY clause along with the ORDER BY and LIMIT clauses.

**Method 1: Using GROUP BY with ORDER BY (MySQL)**

****

**Method 2: Using Subquery (MySQL, SQL Server)**

****

**Conclusion:**

* Use GROUP BY with ORDER BY for a simple approach.
* Use Subqueries for better performance with large datasets.

**Optimization Techniques**



## How to Optimize SQL Queries?

Optimizing SQL queries improves performance and execution speed while handling large datasets.

## Best Practices to Optimize SQL Queries:

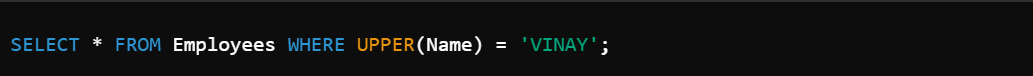
* 1. **Use Indexes:** Create indexes on columns used in WHERE, JOIN, and ORDER BY clauses.
  2. **Avoid SELECT:** Select only required columns instead of using SELECT
  3. **Use Joins Efficiently:** Use INNER JOIN instead of OUTER JOIN whenever possible.
  4. **Use EXISTS Instead of IN:** EXISTS performs better with large datasets**.**

****

* 1. **Use LIMIT or TOP:** Fetch only required rows using LIMIT or TOP.



* 1. **Avoid Functions in WHERE Clause:**

**Instead of:**

**Use:**



* 1. **Optimize Subqueries:** Use JOIN instead of subqueries whenever possible.
  2. **Partition Large Tables:** Split large tables into smaller partitions.
  3. **Use CTEs and Temp Tables:** Store temporary results for better performance.
  4. **Analyze Execution Plan:**Use EXPLAIN or Query Plan to check how queries

are executed.

## What is Query Execution Plan?

A Query Execution Plan is a detailed roadmap used by the database engine to execute SQL queries efficiently.

# Key Points:

* Shows how the database will retrieve data.
* Helps to analyze performance and optimize queries.
* Displays the order of execution for each query operation like joins, scans, and sorting.
* Provides information about indexes, table scans, and filtering methods.

**How to View Execution Plan:**

**MySQL:**

****

**SQL Server:**



# Why Use Execution Plan?

* Identify performance bottlenecks.
* Check if the query is using Indexes or Table Scans.
* Understand how Joins and Filters are applied.

## How to Improve Query Performance?

Improving Query Performance ensures faster data retrieval and better database efficiency, especially for large datasets.

## Best Practices to Improve Query Performance:

1. **Use Indexes:** Create indexes on columns used in WHERE, JOIN, and ORDER BY clauses.
2. **Avoid SELECT:** Select only required columns instead of using SELECT
3. **Use Joins Efficiently:** Prefer INNER JOIN over OUTER JOIN when possible.
4. **Use WHERE Instead of HAVING:** Filter rows early using WHERE.



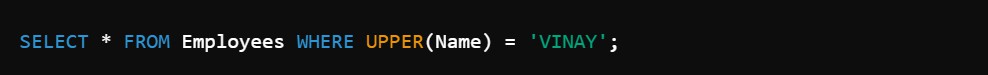
1. **Limit Results:** Fetch only the required number of rows.



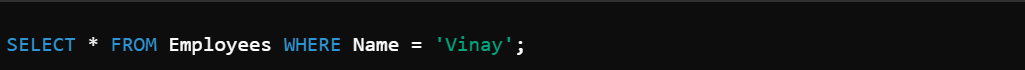
1. **Use EXISTS Instead of IN:** EXISTS performs better with subqueries.



1. **Avoid Functions in WHERE Clause:**

**Instead of:**

**Use:**

****

1. **Partition Large Tables:** Split large tables into smaller partitions.
2. **Use Temporary Tables or CTEs:** Store temporary results for better performance.
3. **Analyze Execution Plans:** Use EXPLAIN or SHOWPLAN to understand query execution.

## What is Indexing?

Indexing in SQL is a technique used to improve the speed of data retrieval from a database by creating a lookup table for faster access.

## Key Points:

* Works like a book index to find data quickly.
* Reduces the time required for SELECT queries.
* Automatically updated when data is inserted, updated, or deleted.
* Indexes are created on columns frequently used in WHERE, JOIN, and ORDER BY clauses.

## Types of Indexes:

1. **Primary Index –** Automatically created on Primary Key columns.
2. **Unique Index –** Ensures that values in a column are unique.
3. **Clustered Index –** Sorts data rows physically based on key values (Only one per table).
4. **Non-Clustered Index –** Stores pointers to data rows (Multiple indexes allowed).

**Conclusion:** Indexing improves query performance by quickly locating data but may increase the time for INSERT, UPDATE, and DELETE operations.

## What is Table Partitioning?

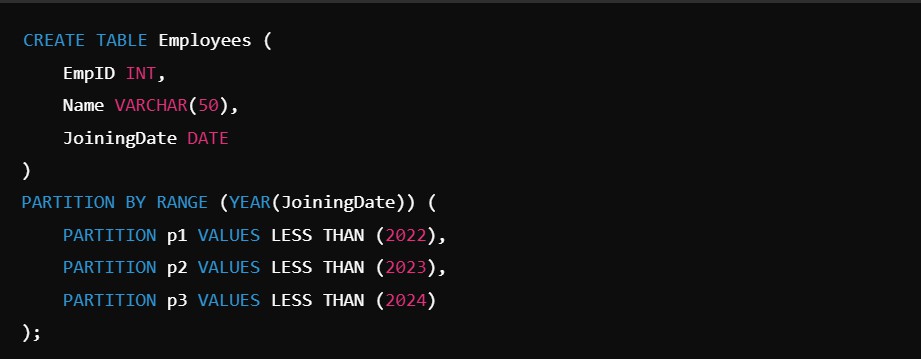
Table Partitioning is a technique used to divide large tables into smaller, more manageable pieces without changing the table structure.

## Key Points:

* Improves query performance on large datasets.
* Simplifies data management.
* Helps in faster data retrieval.
* Each partition is stored separately.
* Data can be partitioned by range, list, hash, or composite methods.

## Types of Partitioning:

1. **Range Partitioning –** Divides data based on value ranges.
2. **List Partitioning –** Divides data based on specific column values.
3. **Hash Partitioning –** Distributes data evenly using a hash function.
4. **Composite Partitioning –** Combination of Range and Hash partitioning.

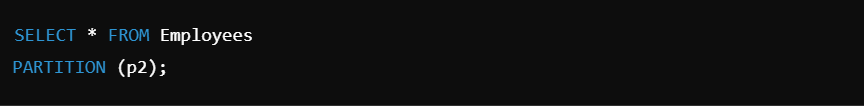


**Syntax (MySQL Range Partitioning):**



Allow non-blocking reads for read-only operations.

## Example Query:

****

**Conclusion:** Table Partitioning improves performance and data management by splitting large datasets into smaller, more manageable sections.

## How to Avoid Deadlocks in SQL?

A Deadlock occurs when two or more transactions block each other by holding locks on resources that the other transactions need.

## Ways to Avoid Deadlocks:

* 1. **Access Tables in the Same Order:**
     + Always access tables in a consistent sequence across transactions.

## Minimize Lock Time:

* + - Keep transactions short and fast to reduce lock holding time.

## Use Lower Isolation Levels:

* + - Use READ COMMITTED instead of SERIALIZABLE isolation level if possible.

## Avoid User Interaction Inside Transactions:

* + - Do not wait for user input during a transaction.

## Use NOLOCK or Read Uncommitted:

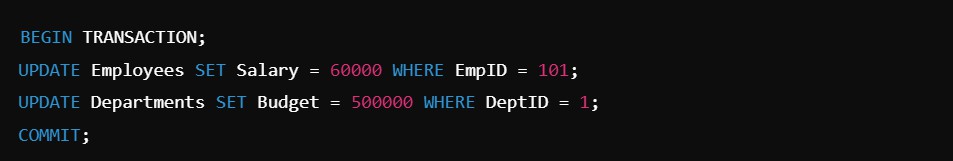
* 1. **Proper Indexing:**

Use Indexes to minimize the number of rows locked.

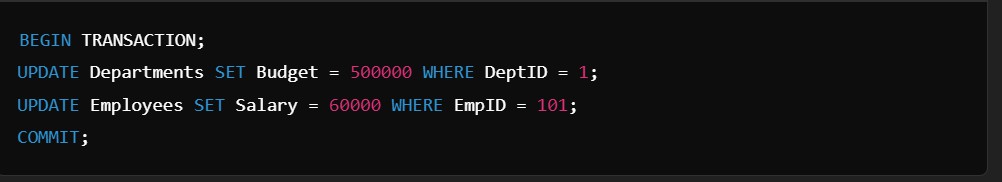
## Break Large Transactions:

Split large transactions into smaller batches.

**Example:**

**Without Deadlock Prevention:**

**With Deadlock Prevention:**

****

**Conclusion:**

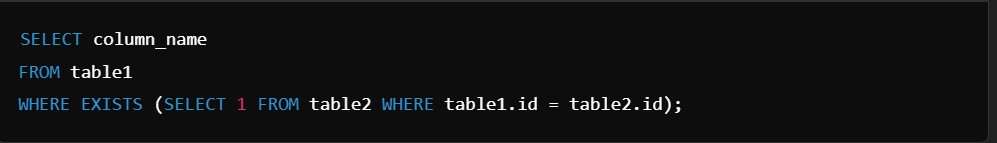
Follow consistent table access patterns, minimize lock times, and use proper indexing to avoid deadlocks in SQL transactions.

## What is the use of EXISTS in SQL?

EXISTS in SQL is used to check whether a subquery returns any rows. It returns:

* TRUE if the subquery returns at least one row.
* FALSE if the subquery returns no rows.

## Syntax:

****

**Conclusion:** EXISTS is faster than IN for large datasets and is commonly used in subqueries to check data existence.

## What is Query Optimization?

Query Optimization is the process of improving the efficiency and performance of SQL queries to retrieve data faster while using minimal system resources.

## Key Points:

* Helps in reducing query execution time.
* Improves database performance.
* Minimizes CPU usage, memory usage, and I/O operations.
* Automatically performed by the Query Optimizer in most RDBMS.

## Techniques for Query Optimization:

1. Use Indexes to speed up searches.
2. Avoid SELECT and select only necessary columns.
3. Use JOINs instead of subqueries when possible.
4. Use EXISTS instead of IN for large datasets.
5. Avoid Functions in WHERE clause.
6. Use LIMIT or TOP to fetch only required rows.
7. Analyze the Execution Plan using tools like EXPLAIN.

## Conclusion:

Query Optimization improves execution speed, reduces resource usage, and enhances overall database performance.

1. **What is the Difference Between Stored Procedure and Function in SQL?**

**Stored Procedure**

Can return multiple values.

Can perform DML operations like INSERT, UPDATE, DELETE.

Supports input and output parameters. Can call Functions inside it.

Used for business logic and complex

operations.

**Function**

Returns only one value (scalar or table).

Cannot perform DML operations.

Only supports input parameters. Cannot call Stored Procedures inside it.

Used for calculations and returning values.

1. **What is Query Optimization?**

OLTP (Online Transaction Processing)

Used for day-to-day transactional operations.

Focuses on data consistency and speed.

Stores detailed transactional data.

Supports INSERT, UPDATE, DELETE

operations.

Small amounts of data processed per transaction.

Example: Banking systems, E-commerce websites.

OLAP (Online Analytical Processing)

Used for data analysis and reporting.

Focuses on data aggregation and analysis.

Stores historical and summarized data.

Supports SELECT operations with complex queries.

Large amounts of data processed at once.

Example: Data Warehouses, Business Intelligence Tools.