SQL Server Question:

01: Basic

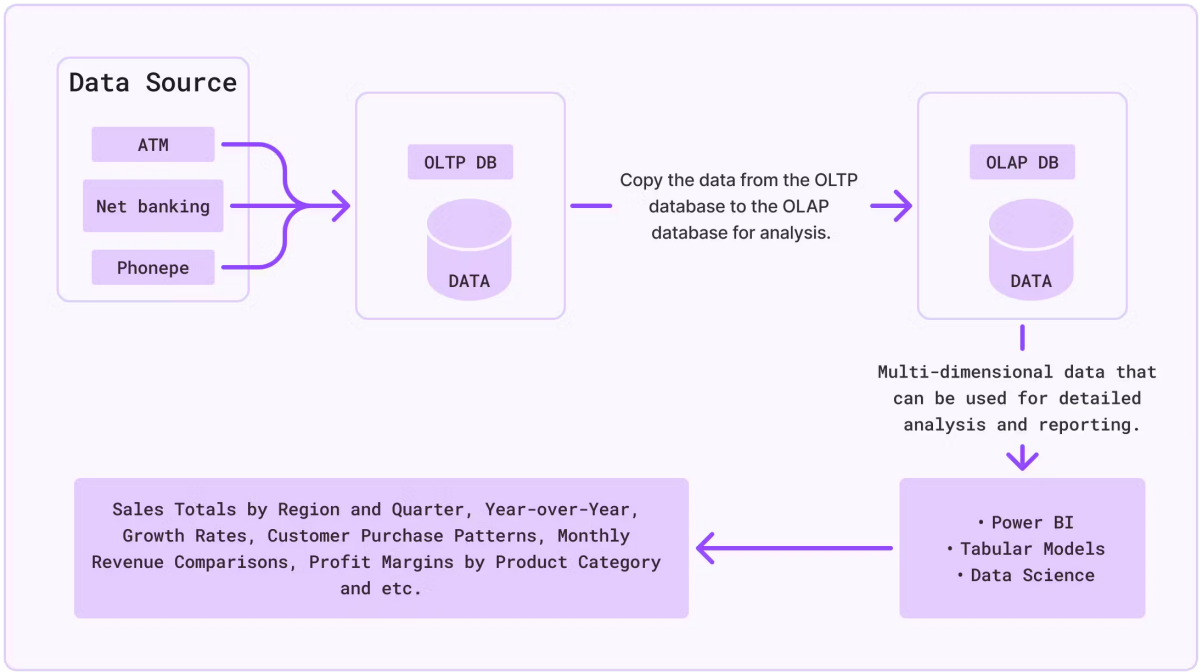
- ▶ What is SQL Server?
- ▶ What is Database (DB)

Types of Databases

- ▶ What are the two main types of databases?
- ▶ What is the key difference between OLTP and OLAP databases?
- ▶ What are the basic day-to-day operations on a database?
- ▶ How do organizations use OLTP and OLAP databases?

Difference between OLTP and OLAP Databases

- ▶ Diagram?

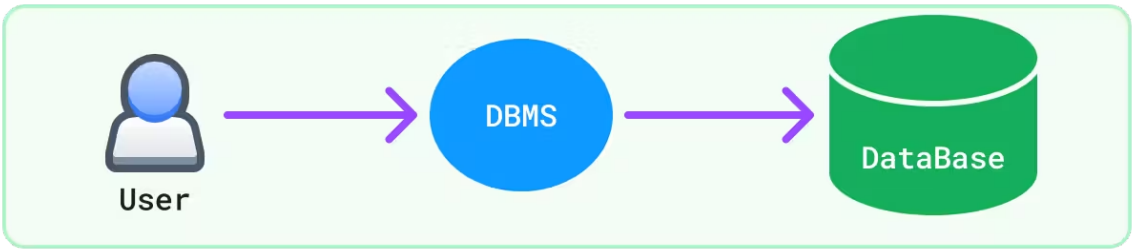


- ▶ What is the main purpose of OLTP and OLAP databases?
- ▶ How do OLTP and OLAP databases differ in their data operations?
- ▶ What type of data volume do OLTP and OLAP databases handle?
- ▶ What is the difference in data structure between OLTP and OLAP databases?
- ▶ What types of queries are used in OLTP and OLAP databases?
- ▶ How is performance optimized in OLTP and OLAP databases?
- ▶ Can you give examples of OLTP and OLAP databases?
- ▶ Who are the typical users of OLTP and OLAP databases?
- ▶ What is the nature of transactions in OLTP and OLAP databases?
- ▶ How critical is data integrity in OLTP and OLAP databases?

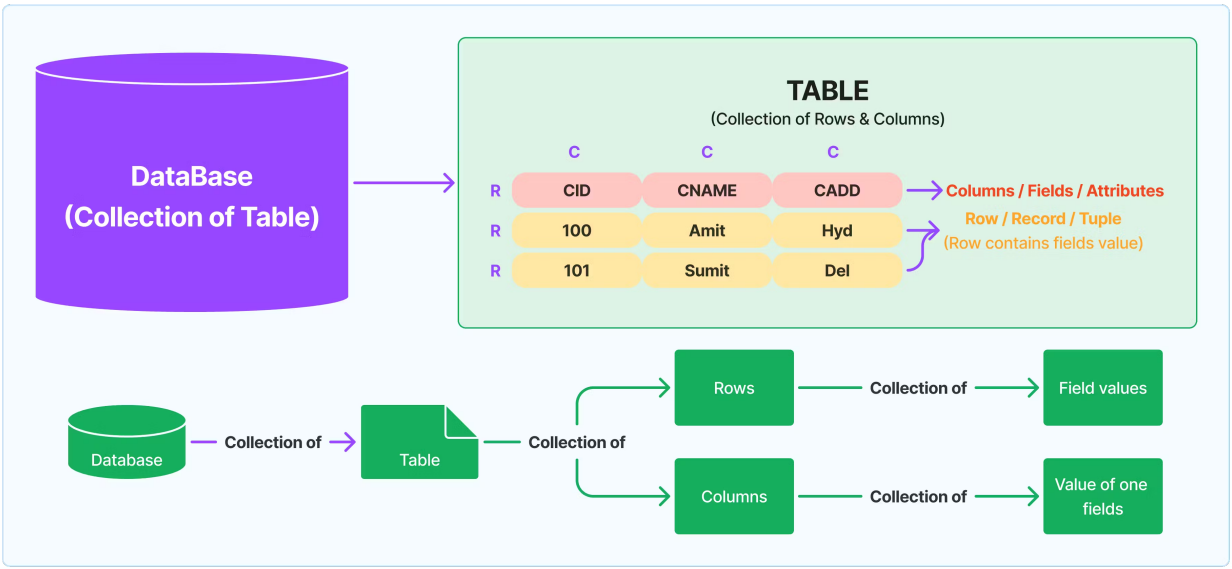
- ▶ What is the typical response time for OLTP and OLAP databases?

Database Management Systems (DBMS)

- ▶ What is a Database Management System (DBMS)?
- ▶ Digram?



- ▶ What are the main types of DBMS?
- ▶ What is RDBMS (Relational Database Management System)?
- ▶ Give some important rules of 12 Codd's rules of an RDBMS
- ▶ Diagram?

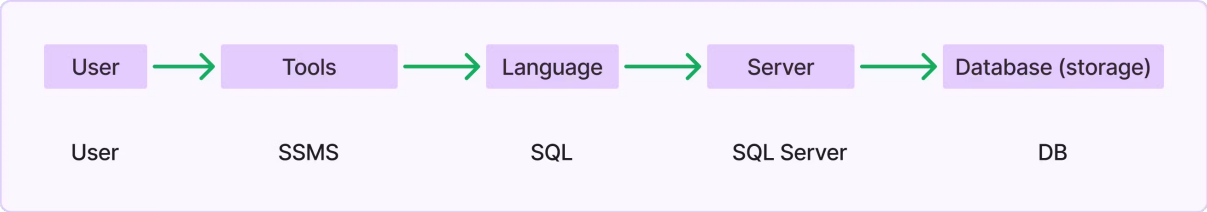


- ▶ What are the key features of an RDBMS?
- ▶ What are the ACID properties in RDBMS?
- ▶ What are some examples of RDBMS software?
- ▶ What are NoSQL databases?
- ▶ What is ORDBMS (Object Relational Database Management System)?
- ▶ What is SQL Server?
- ▶ What are some examples of databases?

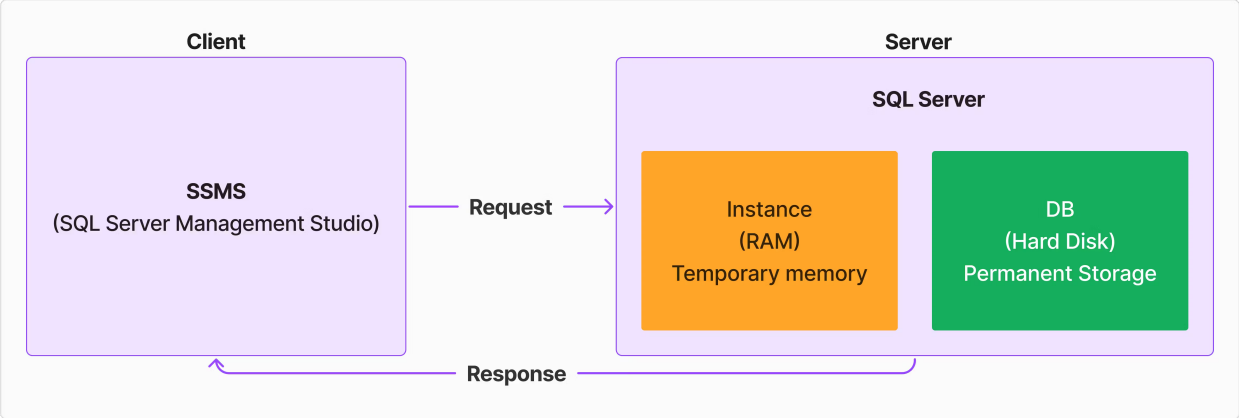
Database Development Life Cycle and SQL Server

- ▶ What is the Database Development Life Cycle?
- ▶ What are the roles of a Developer and a DBA in the database development process?
- ▶ DBA's or Developer, who can Access, Read and update?

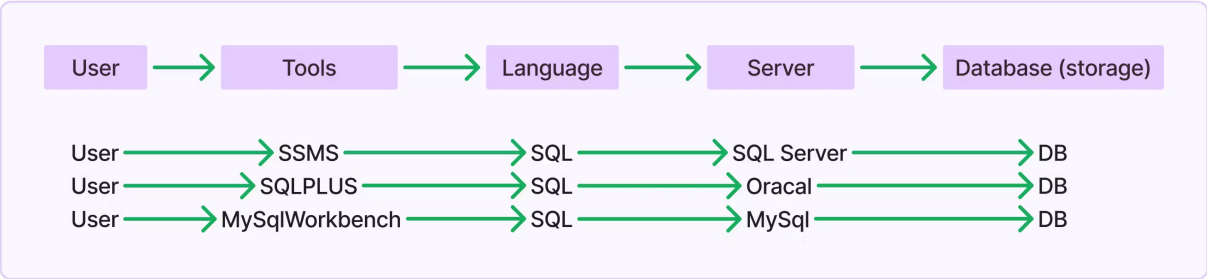
- ▶ What are the different versions of SQL Server?
- ▶ What is the Client-Server Architecture in SQL Server?



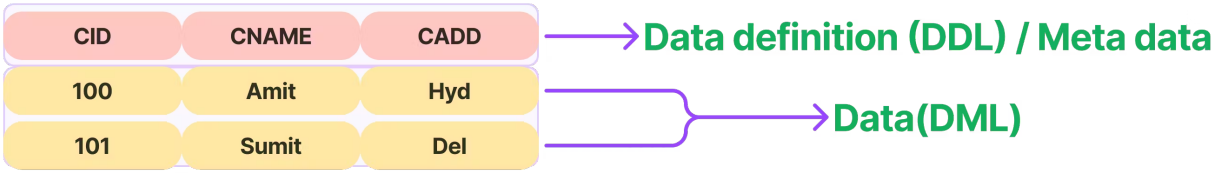
- ▶ Diagram?



- ▶ What is SQL (Structured Query Language)?



- ▶ What are the sub-languages of SQL and their purposes?
- ▶ What are the components of SQL Server?
- ▶ What is SQL Server Management Studio (SSMS)?
- ▶ What is the origin and commonality of SQL?
- ▶ what is data and data defination?



SQL Server Database Creation, Data Types, and Table Management

- ▶ How can you create a new database in SQL Server using SQL Server Management Studio (SSMS)?
- ▶ How can you create a new database using a SQL query?

```
--Syntax:
CREATE DATABASE <database_name>;

--Example:
CREATE DATABASE <Company>;
```

```
```

- ▶ What are the database file?
- ▶ Datatype Diagram?

ASCII (CHAR)	UNICODE (CHAR)	INTEGER	FLOAT	CURRENCY	DATE	BINARY
char	nchar	tinyint	numeric(p,s)	smallmoney	date	binary
varchar	nvarchar	smallint	decimal(p,s)	money	time	varbinary
varchar(max)	nvarchar(max)	int			datetime	varbinary(max)
		bigint				
		numeric(p)				

- ▶ What is the difference between CHAR, VARCHAR, and VARCHAR(MAX)?
- ▶ What types of characters can CHAR, VARCHAR, and VARCHAR(MAX) store?
- ▶ Can CHAR, VARCHAR, and VARCHAR(MAX) store characters from all languages?
- ▶ What is the difference between NCHAR(size), NVARCHAR(size), and NVARCHAR(MAX) in SQL Server?
- ▶ What are the differences between TINYINT, SMALLINT, INT, and BIGINT in SQL Server?
- ▶ What is the FLOAT data type in SQL Server?
- ▶ What is the NUMERIC(p) or DECIMAL(p) data type in SQL Server?

```
empid NUMERIC(4)   -- 4 is the number of digits
-- Valid values:
-- 10 - ok
-- 100 - ok
-- 1000 - ok
-- 10000 - not ok (exceeds 4 digits)

phone NUMERIC(10)
aadharno NUMERIC(12)
```

- ▶ What is the difference between NUMERIC(p, s) and DECIMAL(p, s)?

```
salary NUMERIC(7,2); -- Total number of digits is 7, with 2 digits after the
decimal point.
salary DECIMAL(7,2);

-- Valid values:
-- 5000 - ok
-- 5000.55 - ok
-- 50000.55 - ok
-- 500000.55 - not ok (exceeds 7 digits in total)
-- 5000.5689 - ok because it's rounded to 5000.57
```

- ▶ What are the key features of NUMERIC(p, s) or DECIMAL(p, s)?

- ▶ What are SMALLMONEY and MONEY data types in SQL Server?
- ▶ What is the difference between SMALLMONEY and MONEY in SQL Server?

```
salary SMALLMONEY   -- Add when creating table
balance MONEY       -- Add when creating table
```

- ▶ What are the DATE, TIME, and DATETIME data types in SQL Server?
- ▶ What are the default formats for DATE and TIME in SQL Server?

```
dob DATE  -- Add when creating table, e.g., '2003-03-10' is the format.
login TIME -- Add when creating table, e.g., '09:30:00' is the format.
signup DATETIME -- Add when creating table, e.g., '2023-09-13 10:00:00' is
the format.
```

Command's:-

- ▶ What is the SQL command to create the `Company` database? 🐞

```
--Syntax:
CREATE DATABASE <Database_Name>;

--Example:
CREATE DATABASE Company;
```

- ▶ How do you create a table in SQL Server? 🐞

```
--Syntax:
CREATE TABLE <Table_Name>(
  <Field_Name> <DataType_Size>,
  <Field_Name> <DataType_Size>,
  ...
);
--Example:
CREATE TABLE emp (
    Gender CHAR(1),
    FullName VARCHAR(100),
    AboutUs VARCHAR(MAX),
    FirstName NCHAR(50),
    LastName NCHAR(50),
    ProductName NVARCHAR(100),
    AboutProduct NVARCHAR(MAX),
    AgeGroup TINYINT,
    QuantityOnHand SMALLINT,
    EmployeeID INT,
```

```
    TransactionID BIGINT,
    MonthlySalary NUMERIC(7, 2),
    Incentive SMALLMONEY,
    Package MONEY,
    JoiningDate DATE,
    OfficeComingTime TIME,
    DOB DATETIME
);
```

- ▶ Create a propper employee table for Company?

```
Create Table Employee
(
    Id int,
    FirstName Varchar(30),
    LastName Varchar(30),
    FullName NVARCHAR(100),
    Address NVARCHAR(MAX),
    DOB DATETIME,
    Gender CHAR,
    Salary NUMERIC(7,2),
    Incentive SMALLMONEY,
    JoinDate DATE,
);
```

- ▶ What are the rules for naming tables in SQL Server?
- ▶ What is the SP_HELP command in SQL Server?

```
--Syntax:-
SP_HELP <tablename>
--Example:-
SP_HELP emp;
```

Output:-

Column Name: The name of each column in the table.

Data Type: The data type of each column (e.g., tinyint, varchar, smallmoney).

Length: The maximum storage size for that column.

Nullable: Indicates if the column can contain a NULL value (Yes or No).

- ▶ How do you insert data into a table in SQL Server?

```
INSERT INTO Employee VALUES (1, 'John', 'Doe', 'N John Doe', '123 Elm
Street, Springfield', '1985-06-15', 'M', 5000.50, 200.00, '2010-03-01');
```

For multiple rows with data function's:

```
INSERT INTO Employee VALUES
(2, 'Jane', 'Smith', 'N Jane Smith', 'N 456 Maple Avenue, Riverdale',
DATEADD(YEAR, -33, GETDATE()), 'M', 4200.75, 150.00, DATEADD(YEAR, -8,
GETDATE())),
(3, 'Michael', 'Johnson', 'N Michael Johnson', 'N 789 Oak Lane, Brookfield',
DATEADD(YEAR, -35, GETDATE()), 'F', 5500.00, 300.00, DATEADD(YEAR, -12,
GETDATE())),
(4, 'Emily', 'Clark', 'N Emily Clark', 'N 321 Pine Street, Lakeview',
DATEADD(YEAR, -30, GETDATE()), 'F', 4700.25, 180.00, DATEADD(YEAR, -6,
GETDATE()));
```

- ▶ How to insert NULL values into the table? 🐞

```
INSERT INTO Employee VALUES (5, 'Olivia', 'Brown', NULL, '202 Birch Street,
Springfield', NULL, 'M', 54000.00, NULL, DATEADD(YEAR, -6, GETDATE()));
-- OR
INSERT INTO Employee(Id, FirstName, FullName, Address, Gender, Incentive,
JoinDate) VALUES (6, 'Liam', 'Liam Jones', '303 Cedar Drive, Springfield',
'F', 220.00, '2020-11-30'); -- And other value shood automaticly be null
```

- ▶ How to display all data?

```
SELECT * FROM Employee;
```

  - means all columns.

- ▶ How to display specific columns (FullName, Address, and Salary)?

```
SELECT FullName, Address, Salary FROM Employee;
```

- ▶ What are arithmetic operators in SQL Server?
- ▶ What are relational operators in SQL Server?
- ▶ What are logical operators in SQL Server?
- ▶ What are special operators in SQL Server?
- ▶ What are set operators in SQL Server?

- ▶ What are the Clause in SQL?
- ▶ What is the purpose of the WHERE clause in SQL Server?

```
SELECT columns
FROM tabname
WHERE <condition>;
```

- ▶ How do you structure a condition in a WHERE clause?

```
COLNAME OP VALUE
```

  - **COLNAME:** The column name to apply the condition to.
  - **OP:** A relational operator (e.g., >, >=, <, <=, =, <>).
  - **VALUE:** The value to compare against.

- ▶ Can you provide examples of using the WHERE clause?

1. Display employee details where Id = 3:

```
SELECT *
FROM Employee
WHERE Id = 3;
```

2. Display employee details where FullName = 'N Emily Clark':

```
SELECT *
FROM Employee
WHERE FullName = 'N Emily Clark';
```

3. Display employee details earning more than 5000:

```
SELECT *
FROM Employee
WHERE Salary > 5000;
```

4. Display employees who joined after 2020:

```
SELECT *
FROM Employee
```

```
WHERE JoinDate > '2020-01-01'
```

5. Display employees who joined before 2020:

```
SELECT *
FROM Employee
WHERE JoinDate < '2020-01-01'
```

- ▶ What is a compound condition in a WHERE clause?
- ▶ Can you provide examples of compound conditions?

```
SELECT *
FROM Employee
WHERE Working = 'clerk' OR Working = 'Manager';
```

2. Employees whose id is 100 or 103:

```
SELECT *
FROM Employee
WHERE Id = 100 OR Id = 103;
```

3. Display male employees older than 30:

```
SELECT *
FROM Employee
WHERE Gender = 'F' AND AGE > 30;
```

4. Employees who joined in the year 2020:

```
SELECT *
FROM Employee
WHERE JoinDate > '2020-01-01' AND JoinDate < '2020-12-31';
```

5. Employees earning more than 5000 and less than 10000:

```
SELECT *
FROM Employee
WHERE Salary > 5000 AND Salary < 10000;
```

- ▶ What is the IN operator in SQL Server? 🐞
  - Invalid syntax using = for multiple values:

```
WHERE COLNAME = V1, V2, V3; -- INVALID👎
```

  - Valid syntax using IN for multiple values:

```
WHERE COLNAME IN (V1, V2, V3, ...); --VALID👍
```

- ▶ Could you provide the Example of IN operator with Where close? 🐞

```
SELECT *
FROM Employee
WHERE Id IN (100, 103, 105);
```

2. Employees working as a clerk, manager, or analyst:

```
SELECT *
FROM Employee
WHERE Working IN ('clerk', 'manager' , 'analyst');
```

3. Employees not working as a clerk or manager:

```
SELECT *
FROM Employee
WHERE Working NOT IN ('clerk', 'manager');
```

- ▶ What is the BETWEEN operator in SQL? 🐞

Note: Use the BETWEEN operator with the lower value first and the upper value second to ensure correct results.

Syntax:

```
WHERE COLNAME BETWEEN V1 AND V2;
```

- ▶ Could you provide the example of bitween operators with wgere cloase? 🐞

```sql
SELECT *
FROM EMP
WHERE SAL BETWEEN 5000 AND 10000;
```

- ▶ What happens when you use BETWEEN with the range in reverse?

```sql
SELECT *
FROM EMP
WHERE SAL BETWEEN 10000 AND 5000;
```

Options:
A: ERROR
B: RETURNS ROWS
C: RETURNS NO ROWS
D: NONE

The correct answer is C: RETURNS NO ROWS because BETWEEN 10000 AND 5000 is logically equivalent to (SAL >= 10000 AND SAL

Note: Use the BETWEEN operator with the lower value first and the upper value second to ensure correct results.

- ▶ What are some practical examples of using these operators together?

```sql
SELECT *
FROM Employee
WHERE Working IN ('clerk', 'manager')
      AND Salary BETWEEN 5000 AND 10000
      AND JoinDate BETWEEN '2023-01-01' AND '2023-12-31'
      AND Gender = 'M';
```

1. List of Samsung and Realme mobile phones priced between 10000 and 20000:

```sql
SELECT *
FROM PRODUCTS
WHERE BRAND IN ('SAMSUNG', 'REALME')
  AND CATEGORY = 'MOBILES'
  AND PRICE BETWEEN 10000 AND 20000;
```

- ▶ What is the LIKE operator in SQL?

Wildcard Characters:

- **%** : Represents zero or more characters.
- _ : Represents a single character.

- ▶ Can you provide examples of using the LIKE operator? 🐞

```
SELECT *
FROM Employee
WHERE FullName LIKE 's%';
```

2. Employees whose names end with 'd':

```
SELECT *
FROM Employee
WHERE FullName LIKE '%d';
```

3. Employees whose name contains 'a':

```
SELECT *
FROM Employee
WHERE FullName LIKE '%a%';
```

4. Employees where 'a' is the 2nd character in their name:

```
SELECT *
FROM Employee
WHERE FullName LIKE '_a%';
```

5. 'a' is the 3rd character from the last:

```
SELECT *
FROM Employee
WHERE FullName LIKE '%s__';
```

6. Names containing exactly 4 characters:

```
SELECT *
FROM Employee
WHERE FullName LIKE '____';
```

7. Names starting with 'a', 'r', or 'v':

```
SELECT *
FROM Employee
WHERE FullName LIKE '[arv]%';
```

8. Names starting between 'a' and 'p':

```
SELECT *
FROM Employee
WHERE FullName LIKE '[a-p]%';
```

9. Employees who joined in October (yyyy-mm-dd):

```
SELECT *
FROM Employee
WHERE FullName LIKE '_____10___';
```

10. Employees who joined in the year 2020:

```
SELECT *
FROM Employee
WHERE FullName LIKE '2020%';
```

11. Names containing an underscore (_):

```
SELECT *
FROM Employee
WHERE FullName LIKE '%_%';
```

12. Names containing a literal underscore (_), using escape character:

```
SELECT *
FROM Employee
WHERE FullName LIKE '%\_%' ESCAPE '\';
```

13. Names containing a percent sign (%), using escape character:

```
SELECT *
FROM Employee
```

```
WHERE FullName LIKE '%\%%';
```

14. Names containing two underscores (__), using escape character:

```
SELECT *
FROM Employee
WHERE FullName LIKE '%\_%\_%';
```

- ▶ What is the IS operator in SQL Server?
  - To find rows where a column has no value (NULL), use IS NULL. Example: Find employees who are not earning a salary:

```
SELECT *
FROM EMP
WHERE SAL IS NULL;
```

  - To find rows where a column has a value (is not NULL), use IS NOT NULL. Example: Find employees who are earning a salary:

```
SELECT *
FROM EMP
WHERE SAL IS NOT NULL;
```

- ▶ What is an alias in SQL?
- ▶ What is the syntax for using an alias?
  - **column_name:** The original name of the column or expression.
  - **alias_name:** The new name for the column in the query results.

Example: Displaying Employee Names and Annual Salaries

```
SELECT ENAME, SAL * 12 AS ANNSAL FROM EMP;
```

Example with Spaces or Special Characters in Alias: Aliases can be enclosed in double quotes if they contain spaces or special characters:

```
SELECT ENAME, SAL * 12 AS "ANNUAL SAL" FROM EMP;
```

- ▶ Can you use an alias in the WHERE clause?

Example: Filtering Using Calculated Column:

```
SELECT *, SAL * 12 AS ANNSAL
FROM EMP
WHERE SAL * 12 > 60000;
```

- ▶ How can you calculate and display multiple columns with aliases?

```
SELECT ENAME, SAL,
       SAL * 0.2 AS HRA,    -- House Rent Allowance (20% of SAL)
       SAL * 0.3 AS DA,     -- Dearness Allowance (30% of SAL)
       SAL * 0.1 AS TAX,    -- Tax (10% of SAL)
       SAL + (SAL * 0.2) + (SAL * 0.3) - (SAL * 0.1) AS TOTSAL --Total
Salary
FROM EMP;
```

- ▶ How do you handle NULL values in calculations in SQL?
- ▶ What is the use of ISNULL and COALESCE and how to use it?

Example: Handling NULL Values Using ISNULL

```
SELECT ISNULL(Salary, 0) FROM Employee;
--With coloum name:
SELECT ISNULL(Salary, 0) AS NewSalary FROM Employee;
```

This query ensures that `if Salary is NULL`, it will be treated as `0` during the calculation of TOTSAL.

2. COALESCE: COALESCE is another function available in SQL Server that returns the first non-NULL value from a list of arguments (To replace NULL values in the Incentive column with 0).

Example: Handling NULL Values Using COALESCE

```
SELECT Id, FullName, Address, DOB, Gender,Salary,
 COALESCE(Incentive, 0) AS TotalPay
FROM Employee;
```

- ▶ Diffrence bitween ISNULL and COALESCE?

```
SELECT FullName, Address, DOB, Gender, Salary,
  COALESCE(
    CONVERT(VARCHAR, DOB),
    CONVERT(VARCHAR, Salary),
```

```
        'No Contact Available')
  AS FirstAvailableContact
FROM Employee;
```

Explanation:

- CONVERT(VARCHAR, DOB): Converts the DOB (Date of Birth) to a VARCHAR type.
- CONVERT(VARCHAR, Salary): Converts the Salary to a VARCHAR type.
- 'No Contact Available': A default string value if both DOB and Salary are NULL.

- ▶ How do you display total and average scores using aliases?

Example: Displaying Total and Average Scores for Students:

```
SELECT SID,
   S1 + S2 + S3 AS TOTAL,  -- Total score
   (S1 + S2 + S3) / 3 AS AVG -- Average score
FROM STUDENT;
```

Key Takeaways

- Aliases provide a way to give temporary names to columns or tables in SQL queries for better readability.
- The NVL function helps handle NULL values by replacing them with a specified value.
- Using aliases allows you to perform complex calculations and display meaningful names in query results.

- ▶ What is the purpose of the ORDER BY clause in SQL?
- ▶ What is the syntax for using the ORDER BY clause?

```
SELECT columns
FROM table_name
[WHERE condition]
ORDER BY column1 [ASC|DESC], column2 [ASC|DESC], ...;
```

- column1: The column used for the first level of sorting.
- column1: The column used for the first level of sorting.

Note

- First Level of Sorting (DEPTNO ASC): Sorts all employees by the DEPTNO column in ascending order.
- Second Level of Sorting (SAL DESC): Within each department (group of rows with the same DEPTNO), sorts the rows by the SAL column in descending order.

- ▶ Can you provide examples of using the ORDER BY clause?

```
SELECT * FROM EMP
ORDER BY ENAME ASC;
```

2. Arranging Employees by Salary in Descending Order:

```
SELECT * FROM EMP
ORDER BY ENAME ASC;
```

3. Arranging Employees by Department (Ascending) and Within Department by Salary (Descending):

```
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP
ORDER BY DEPTNO ASC, SAL DESC;
```

- ▶ How does the sorting process work in multiple column sorting? 🐞

1. **First Level of Sorting:** The database sorts the results by the first column.
2. **Subsequent Levels of Sorting:** For rows where the values in the first column are identical, it sorts by the next specified column, and so on.

Example: Arranging Employees by Department and Hire Date:

```
SELECT EMPNO, ENAME, HIREDATE, DEPTNO
FROM EMP
ORDER BY DEPTNO ASC, HIREDATE ASC;
```

- First, it sorts by DEPTNO in ascending order.
- Within each department, it sorts by HIREDATE in ascending order.

- ▶ How do you arrange students by average score in descending order and then by marks in specific subjects?
- ▶ Can you example: Arranging Students by Average Score, Marks in Math (Mth), and Physics (Phy)?

```
SELECT SNO, SNAME, Mth, Phy, Che, (Mth + Phy + Che) / 3 AS AVG
FROM STUDENT
ORDER BY AVG DESC, Mth DESC, Phy DESC;
```

Explanation:

- Sorts students by their average score (AVG) in descending order.

- If two students have the same average score, sorts them by marks in Math (Mth) in descending order.
- If two students have the same marks in Math, sorts them by marks in Physics (Phy) in descending order.

- ▶ How can you arrange employees working as a 'Clerk' or 'Manager' by salary in descending order?

```
SELECT EMPNO, ENAME, JOB, SAL
FROM EMP
WHERE JOB IN ('CLERK', 'MANAGER')
ORDER BY SAL DESC;
```

- ▶ What are the key points to remember about the ORDER BY clause?
  1. **Multiple Columns for Sorting**: You can include multiple columns for sorting. The order of columns determines the priority of sorting.
  2. **Case-Sensitivity**: Sorting in SQL can be case-sensitive, depending on the database's collation settings.
  3. **Execution Order**:
     - **Aliases in WHERE Clause**: Aliases in ORDER BY Clause
     - **Aliases in ORDER BY Clause**: Aliases can be used in the ORDER BY clause because it is executed after SELECT.

- ▶ Can aliases be used in the ORDER BY clause?

```
SELECT Salary, Salary * 0.20 AS TotalSalary  -- Alias for a calculated
column
FROM Employee
ORDER BY Incentive ASC;-- Using the alias in ORDER BY
```

- ▶ What is the purpose of the DISTINCT clause in SQL?
- ▶ What is the syntax for using the DISTINCT clause?

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

- `column1, column2, ...`: The columns for which you want to ensure unique values.

- ▶ Can you provide some examples of using the DISTINCT clause?

```
SELECT DISTINCT JOB
FROM EMP;
```

Result

JOB
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN

This query returns all unique job titles from the EMP table, removing any duplicates.

- ▶ Selecting Distinct Department Numbers from the Employee Table:

```
SELECT DISTINCT DEPTNO
FROM EMP;
```

This query retrieves all unique department numbers (DEPTNO) from the EMP table.

- ▶ Selecting Distinct Combinations of Department Number and Job Title? 🐞

```
SELECT DISTINCT DEPTNO, JOB
FROM EMP;
```

Result

DEPTNO	JOB
10	CLERK
10	MANAGER
10	PRESIDENT
20	SALESMAN
20	ANALYST
20	CLERK
30	ANALYST
30	CLERK
30	SALESMAN

This query returns all unique combinations of department numbers (DEPTNO) and job titles (JOB) from the EMP table.

- ▶ How does the DISTINCT clause handle multiple columns?

```
SELECT DISTINCT DEPTNO, JOB
FROM EMP;
```

This query returns only rows where the combination of `DEPTNO` and `JOB` is unique.

- ▶ Are there any performance considerations when using the DISTINCT clause?
- ▶ Can DISTINCT be used with other SQL clauses like ORDER BY or GROUP BY?
  - `ORDER BY:` You can use ORDER BY to sort the unique results.

```
SELECT DISTINCT JOB
FROM EMP
ORDER BY JOB ASC;
```

  - `GROUP BY:` You can use DISTINCT along with GROUP BY for more complex data grouping.

```
SELECT DISTINCT DEPTNO, COUNT(*)
FROM EMP
GROUP BY DEPTNO;
```

- ▶ What happens if you use DISTINCT with all columns?

```
SELECT DISTINCT *
FROM EMP;
```

This query returns all unique rows in the EMP table, considering all columns.

- ▶ What is the purpose of the TOP clause in SQL?
- ▶ What is the syntax for using the TOP clause?

```
SELECT TOP n *
FROM table_name
ORDER BY column_name;
```

  - `n:` The number of rows to return.
  - `column_name:` The column used to order the data.

- ▶ Can you provide examples of using the TOP clause?

```
SELECT TOP 3 *
FROM EMP;
```

This query returns the first 3 rows from the `EMP` table based on the default order of rows in the table.

2. Display the Top 3 Highest Paid Employees:

```
SELECT TOP 3 *
FROM EMP
ORDER BY SAL DESC;
```

This query returns the top 3 employees with the highest salaries by sorting the rows in descending order based on the SAL (salary) column.

3. Display the Top 3 Employees Based on Experience:

```
SELECT TOP 3 *
FROM EMP
ORDER BY HIREDATE ASC;
```

This query returns the top 3 employees based on their hire date, sorted in ascending order. This effectively shows the employees who have been with the company the longest.

4. Display the Top 4 Maximum Salaries:

```
SELECT DISTINCT TOP 4 SAL
FROM EMP
ORDER BY SAL DESC;
```

This query returns the top 4 unique highest salaries from the `EMP` table.

- ▶ Is the TOP clause supported by all SQL databases?
  - MySQL: Use LIMIT instead.

```
SELECT * FROM EMP ORDER BY SAL DESC LIMIT 3;
```

  - Oracle: Use ROWNUM or FETCH FIRST

```
SELECT * FROM EMP WHERE ROWNUM <= 3 ORDER BY SAL DESC;
```

- ▶ How does the TOP clause work with the ORDER BY clause?
- ▶ Can the TOP clause be combined with other SQL clauses?

```
SELECT TOP 2 ENAME, SAL
FROM EMP
WHERE DEPTNO = 10
ORDER BY SAL DESC;
```

This query returns the top 2 highest-paid employees from department number 10.

- ▶ What happens if the TOP clause is used without ORDER BY?
- ▶ Can you use TOP with percentages?

```
SELECT TOP 10 PERCENT *
FROM EMP
ORDER BY SAL DESC;
```

This query returns the top 10% of rows from the EMP table sorted by salary in descending order.

- ▶ Display employee names and their annual salaries using aliases?

```
Select FullName, Salary * 12 as AnualSalary  from Employee;
```

- ▶ Write a SQL query to show employee names along with their total salary (base salary plus 20% bonus) and use aliases to label the calculated columns.

```
select FullName, (Salary * 12) * 0.2 as TotalSalary from Employee;
```

- ▶ Write a query to display employee names and their experience (calculated as the difference between the current date and the hire date), using an alias for the calculated column.

```
Select FullName as EmployeeName, DATEDIFF(YEAR, JoinDate, GETDATE()) AS
Experience from Employee;
```

- ▶ Display all employee details sorted by their salaries in descending order.

```
Select * from Employee Order by Salary DESC;
```

- ▶ Write a SQL query to sort employee records first by department in ascending order and then by hire date in descending order within each department.

```
Select * From Employee
Order By Dep ASC JoinDate DESC;
```

- ▶ Retrieve employee details ordered by job title in ascending order and then by salary in descending order.

```
Select * From Employee
Order by Job ASC Salary DESC;
```

- ▶ Show a list of employees ordered by their names in alphabetical order, and then by hire date in descending order if names are the same.

```
SELECT * FROM Employee
ORDER BY Name ASC, HireDate DESC;
```

- ▶ Work flow of Order By on String?

Workflow of `ORDER BY` on String Data

1. Character-by-Character Comparison:
   - SQL sorts strings character by character from left to right.
   - It compares the first character of each string; if those are the same, it moves to the second character, and so on, until a difference is found or the end of the string is reached.
2. Lexicographical (Dictionary) Order:
   - The default sorting is done in lexicographical or dictionary order, which is based on the character encoding set of the database (like ASCII or Unicode).
   - For example, 'A' comes before 'B', 'B' before 'C', and so on. Lowercase letters ('a', 'b', 'c') typically have higher ASCII or Unicode values than uppercase letters ('A', 'B', 'C'), so 'Apple' comes before 'banana' if case sensitivity is enabled.
3. Case Sensitivity:
   - Sorting can be case-sensitive or case-insensitive, depending on the collation settings of the database.
   - In case-sensitive sorting, 'A' and 'a' are considered different, and uppercase letters are sorted before lowercase ones.
   - In case-insensitive sorting, 'A' and 'a' are considered equal.

4. Whitespace Handling:
   - Spaces and other whitespace characters are considered significant and are sorted before letters or digits.
   - For example, ' Apple' (with leading spaces) would come before 'Apple'.
5. Null Values:
   - If the column contains `NULL` values, they are handled based on the database system.
   - Typically, `NULL` values appear first when sorted in ascending order and last when sorted in descending order. However, this behavior can be modified using `NULLS FIRST` or `NULLS LAST`.
6. Numeric Strings:
   - Strings containing numeric values are sorted as strings, not numbers.
   - For example, '10' comes before '2' because '1' is compared with '2' first, so the order is '1', '10', '2' (lexicographical order).

Example of `ORDER BY` on String Column

Given a table named `Employee` with a `Name` column, the following query:

```
SELECT Name FROM Employee ORDER BY Name;
```

Will perform the following steps:

- **Step 1**: Compare the first character of each name string in the column.
- **Step 2**: If the first characters are the same, compare the second characters, and continue until it finds a difference.
- **Step 3**: Sort the names in ascending order based on lexicographical (dictionary) order.
- **Step 4**: Consider any collation settings for case sensitivity, whitespace, or special characters.

Summary

- **Order By on Strings** sorts based on character encoding values in a lexicographical order.
- **Case sensitivity** and **collation settings** can affect the final output.
- **Whitespace and NULL handling** play a role in determining the sort order.

- ▶ Write a query to list all unique job titles from the employee table.

```
Select DISTINCT Job from Employee;
```

- ▶ Create a query to find distinct department numbers from the employee table.

```
SELECT DISTINCT Did FROM Employee;
```

- ▶ Generate a query to show unique combinations of job title and department number.

```
SELECT DISTINCT Job, Did FROM Employee;
```

- ▶ Retrieve the top 5 highest-paid employees from the employee table.

```
Select TOP 5 * from Employee Order by Salary DESC;
```

- ▶ Show the top 10 employees based on their years of experience (sorted from most experienced to least experienced).

```
SELECT TOP 10 Eid, Ename, HireDate,  DATEDIFF(YEAR, HireDate, GETDATE()) AS
YearsOfExperience
FROM Employee
ORDER BY YearsOfExperience DESC;
```

- ▶ Write a query to get the top 3 employees with the lowest salaries.

```
SELECT TOP 3 Eid, Ename, Salary
FROM Employee
ORDER BY Salary ASC;
```

- ▶ Display the top 4 employees who joined the company most recently.

```
Select Top 4 * from Employee Order by JoinDate ASC
```

- ▶ What is the purpose of using an alias in SQL? Can you provide an example where it improves the readability of a query?

```
SELECT Employee.FirstName + ' ' + Employee.LastName AS FullName,
AVG(Employee.Salary) AS AverageSalary
FROM Employee
GROUP BY Employee.FirstName, Employee.LastName;
```

- ▶ How does using an alias for a column differ from using an alias for a table? Provide an example of each.

```

```

```
SELECT FirstName + ' ' + LastName AS FullName,  -- Column alias
Salary * 1.10 AS SalaryAfterRaise        -- Column alias
FROM Employee;
```

- ▶ How does the ORDER BY clause affect the result set of a query? Can you explain the difference between ascending and descending order?

    1. Sorting in Ascending Order

```
SELECT * FROM Employees
ORDER BY Salary ASC;
```

This query retrieves all employees and sorts them by their salary from the lowest to the highest.

    2. Sorting in Descending Order

```
SELECT * FROM Employees
ORDER BY Salary DESC;
```

This query retrieves all employees and sorts them by their salary from the highest to the lowest.

    3. Sorting by Multiple Columns

```
SELECT * FROM Employees
ORDER BY Department ASC, HireDate DESC;
```

This query first sorts the employees by department in ascending order. Within each department, employees are sorted by their hire date in descending order.

- ▶ If you need to sort a result set by multiple columns, how do you specify the sorting order for each column? Can you provide an example query?

```
SELECT * FROM Employee
ORDER BY FirstName ASC, LastName ASC, FullName DESC, Address DESC, DOB ASC,
Gender ASC, Incentive DESC, JoinDate ASC;
```

- ▶ Explain how the ORDER BY clause can be used in conjunction with other SQL clauses like LIMIT or OFFSET for pagination.

```
  Select * From Employee Where Salary > 4000 Order by JoinDate DESC;
  --or
```

```
  Select * From Employee Where Salary > 4000 AND Salary < 5000 Order by
JoinDate DESC;
```

- ▶ Can you explain a scenario where DISTINCT might be used incorrectly, and what the consequences would be? 🐞

  Certainly! One common scenario where `DISTINCT` might be used incorrectly is when it's applied to a query that includes multiple columns, leading to unexpected results.

  ### Incorrect Use of `DISTINCT`

  Suppose you have an `EMP` table with the following columns: `EmployeeID`, `FirstName`, `LastName`, and `DepartmentNumber`. You want to find unique `DepartmentNumber` values but accidentally include other columns in the `SELECT` statement:

  ```
  SELECT DISTINCT EmployeeID, DepartmentNumber FROM EMP;
  ```

  ### Explanation

  In this query, `DISTINCT` is applied to both `EmployeeID` and `DepartmentNumber`. This means it returns unique combinations of `EmployeeID` and `DepartmentNumber`. If multiple employees belong to the same department, the query still lists each unique employee ID, even though the department numbers might repeat.

  ### Consequences

  - **Incorrect Result Set**: You may end up with more rows than expected, as the query is now filtering distinct pairs of `EmployeeID` and `DepartmentNumber`, not just unique department numbers.
  - **Performance Issues**: Using `DISTINCT` with multiple columns in large tables can slow down query performance, as it requires more data processing.

  ### Correct Usage

  If the goal is to get unique `DepartmentNumber` values, the query should be:

  ```
  SELECT DISTINCT DepartmentNumber FROM EMP;
  ```

  ### Summary

  Using `DISTINCT` incorrectly, such as on unnecessary columns, can lead to misleading results, more rows than intended, and potentially increased query execution time. Always ensure `DISTINCT` is applied to only those columns for which unique values are genuinely required.

- ▶ Explain how to combine the TOP clause with the ORDER BY clause to get the top N rows based on a specific sorting criterion.

```
Select TOP 5 * from Employee Order by DOB ASC;
```

Explanation

  - `Order by DOB ASC`: The rows are sorted first by DOB in descending order.
  - `TOP (5)`: The database then selects the first 5 rows from this sorted list, which will be the top 5 highest salaries.

- ▶ What is DML (Data Manipulation Language)?

  There are 4 DML commands:

  1. **INSERT**
  2. **UPDATE**
  3. **DELETE**
  4. **MARGE**

- ▶ What is the UPDATE Command Used For?
- ▶ What is the Syntax for the UPDATE Command?

```
UPDATE table_name
SET column_name = value, column_name = value, ...
[WHERE condition];
```

  - The WHERE clause is optional and is used to specify which rows to update.

- ▶ Can You Provide Some Examples of the UPDATE Command?
- ▶ When DELETE Command Used?
- ▶ What is the Syntax for the DELETE Command?

```
DELETE FROM table_name [WHERE condition];
```

  - The WHERE clause is optional. Without it, all rows in the table will be deleted.

- ▶ Can You Provide Some Examples of the DELETE Command?
- ▶ Why Should You Use a WHERE Clause with the DELETE Command?
- ▶ How to update all employees' commission to 500?

```
UPDATE EMP SET COMM = 500;
```

- ▶ How to update employees' commission to 500 where commission is NULL?

```
UPDATE EMP
SET COMM = 500
WHERE COMM IS NULL;
```

- ▶ How to update employees' commission to NULL where commission is not NULL?

```
UPDATE EMP
SET COMM = NULL
WHERE COMM IS NOT NULL;
```

- ▶ How to update salary to 2000 and commission to 800 where employee number is 7369?

```
UPDATE EMP
SET SAL = 2000 COMM = 800
WHERE EMPNO = 7369;
```

```
CREATE TABLE Employee
(
    Eid INT PRIMARY KEY,
    Ename VARCHAR(255),
    Job VARCHAR(255),
    Salary DECIMAL(10,2),
    Status BIT,
    Did INT,
    Dname VARCHAR(255),
    Location NVARCHAR(255)
);

INSERT INTO Employee VALUES (101, 'John Doe', 'Manager', 75000.00, 1, 1, 'Human
Resources', 'New York');
INSERT INTO Employee VALUES (102, 'Jane Smith', 'Accountant', 68000.00, 1, 2,
'Finance', 'Chicago');
INSERT INTO Employee VALUES (103, 'Michael Brown', 'Sales Executive', 55000.00, 1,
3, 'Sales', 'Los Angeles');
INSERT INTO Employee VALUES (104, 'Emily Davis', 'Marketing Analyst', 60000.00, 1,
4, 'Marketing', 'Houston');
INSERT INTO Employee VALUES (105, 'William Johnson', 'IT Specialist', 80000.00, 1,
5, 'IT', 'San Francisco');
INSERT INTO Employee VALUES (106, 'Sophia Martinez', 'HR Coordinator', 50000.00,
1, 1, 'Human Resources', 'New York');
```

```
INSERT INTO Employee VALUES (107, 'David Wilson', 'Financial Analyst', 72000.00,
1, 2, 'Finance', 'San Francisco');
INSERT INTO Employee VALUES (108, 'Olivia Garcia', 'Sales Associate', 47000.00, 1,
3, 'Sales', 'Los Angeles');
```

- ▶ How to increase salary by 20% and commission by 10% for salesmen hired in 1981?

```
UPDATE EMP
SET SAL = SAL * 1.2, COMM = COMM * 1.1
WHERE JOB = 'SALESMAN'
AND HIREDATE LIKE '1981%'
```

- ▶ How transfer all employees from department 10 to department 20?

```
UPDATE EMP
SET DEPTNO = 20
WHERE DEPTNO = 10
```

- ▶ How to increase prices by 10% for Samsung, Redmi, and Realme mobile phones?

```
UPDATE PRODUCTS
SET PRICE = PRICE * 1.1
WHERE CATEGORY = 'mobiles'
AND BRAND IN ('samsung', 'redmi', 'realme')
```

- ▶ How to commit the all DML operations?
- ▶ How to stop auto commit DML commands?
- ▶ What is "COMMIT" command?
- ▶ What is "ROLLBACK" command?
- ▶ Can "ROLLBACK" command is posible to undone "COMMIT" command?
  - **Before COMMIT:** If you make changes and have not yet used COMMIT, you can use ROLLBACK to undo those changes.
  - **After COMMIT:** Once you use COMMIT, the changes are saved permanently in the database, and you cannot roll them back.

  So, ROLLBACK can only undo changes if COMMIT has not been executed. After COMMIT, the changes are final.

```
SET IMPLICIT_TRANSACTIONS ON
Update Employee set Salary = 200;
Commit;
Rollback;
```

- ▶ How to cancle the DML operation, execute ROLLBACK?
- ▶ To save a DML (Data Manipulation Language) operation, use the COMMIT command?

Example

```
BEGIN TRANSACTION;
-- Perform some DML operations
UPDATE Employees SET Salary = Salary + 1000 WHERE Department = 'Sales';
-- Save the changes
COMMIT;
```

In this example, the COMMIT statement saves the changes made by the UPDATE operation to the database. If you don't execute COMMIT, you can use ROLLBACK to undo the changes.

BEGIN TRANSACTION:

- This command starts a transaction.
- A transaction is a group of changes you want to make to the database, such as adding, updating, or deleting data.
- Using BEGIN TRANSACTION makes sure that all changes within this group are treated as one unit. If something goes wrong, you can undo all changes at once.

- ▶ What are the Characteristics of DML Commands?
- ▶ What is DDL (Data Definition Language)?
- ▶ What are the Common DDL Commands?
- ▶ What is the ALTER Command Used For?
- ▶ What are Some Uses of the ALTER Command?

```
ALTER TABLE EMP
ADD GENDER CHAR(1);
```

After adding, the new column is filled with NULL by default. To update the new column with data, use the UPDATE command.

```
UPDATE EMP SET GENDER = 'M'
WHERE EMPNO = 7369;
```

- **Dropping Columns**: Removes an existing column from a table.

```
ALTER TABLE EMP
DROP COLUMN GENDER;
```

- **Modifying Columns**: Changes the data type or size of an existing column.

```
ALTER TABLE EMP
ALTER COLUMN EMPNO INT;
```

Increase the size of the ENAME column to 20 characters.

```
ALTER TABLE EMP
ALTER COLUMN ENAME VARCHAR(20);
```

- ▶ What is the DROP Command Used For?

```
DROP TABLE EMP;
```

- ▶ What is the TRUNCATE Command Used For?

```
TRUNCATE TABLE EMP;
```

Use TRUNCATE with caution, as it affects all rows and can lead to data loss.

- ▶ What is the Difference Between DELETE and TRUNCATE?

Feature	DELETE	TRUNCATE
Command Type	DML (Data Manipulation)	DDL (Data Definition)
Deletes Specific Rows	Yes, using a WHERE clause	No, deletes all rows
Deletes All Rows	Yes, if no WHERE clause	Yes, always
Use of WHERE Clause	Allowed	Not allowed
Execution Speed	Slower (row-by-row)	Faster (all rows at once)
Memory Release	No	Yes
Resets Identity Columns	No	Yes

- ▶ What is SP_RENAME Used For?
- ▶ What is the Syntax for SP_RENAME?

```
SP_RENAME 'old_name', 'new_name';
```

  - Example: Rename table EMP to EMPLOYEES

```
SP_RENAME 'EMP', 'EMPLOYEES';
```

- Example: Rename column COMM to BONUS in the EMPLOYEES table.

```
SP_RENAME 'EMPLOYEES.COMM', 'BONUS';
```

- ▶ How to List All Tables Created by the User in the Database?

```
SELECT * FROM INFORMATION_SCHEMA.TABLES;
```

INFORMATION_SCHEMA is a set of read-only views that provide metadata about the database objects, such as tables, columns, and constraints. It is supported by many database systems like SQL Server, MySQL, and PostgreSQL.

- ▶ What are Built-in Functions in SQL Server?
- ▶ What are the types of Built-in Functions in SQL Server:
- ▶ Waht is GETDATE() function?

```
SELECT GETDATE();  -- Output: 2023-09-21 19:15:59.130
```

- ▶ Waht is DATEPART() function?

```
SELECT DATEPART(yy, GETDATE());  -- Year: 2023
SELECT DATEPART(mm, GETDATE());  -- Month: 09
SELECT DATEPART(dd, GETDATE());  -- Day: 21
SELECT DATEPART(dw, GETDATE());  -- 5 (Day of the week)
SELECT DATEPART(dy, GETDATE());  -- 264 (Day of the year)
SELECT DATEPART(hh, GETDATE());  -- Hour
SELECT DATEPART(mi, GETDATE());  -- Minutes
SELECT DATEPART(ss, GETDATE());  -- Seconds
SELECT DATEPART(qq, GETDATE());  -- Quarter: 3
```

- ▶ Display ENAME, SAL, year of join?

```
SELECT ENAME, SAL, DATEPART(yy, HIREDATE) AS YearOfJoin FROM EMP;
```

- ▶ Display employees who joined in 1980, 1983, 1985?

```sql
SELECT * FROM EMP WHERE DATEPART(yy, HIREDATE) IN (1980, 1983, 1985);
```

- ▶ Employees joined in a leap year?

```sql
SELECT * FROM EMP WHERE DATEPART(yy, HIREDATE) % 4 = 0;
```

- ▶ Waht is DATENAME() function?

```sql
SELECT DATENAME(dw, GETDATE());  -- Output: 'Friday'
```

- ▶ Display ENAME and day of the week of hire?

```sql
SELECT ENAME, DATENAME(dw, HIREDATE) AS DayOfWeek FROM EMP;
```

- ▶ Waht is FORMAT() function?

```sql
SELECT FORMAT(GETDATE(), 'dd.MM.yyyy');  -- Output: '22.09.2023'
SELECT FORMAT(GETDATE(), 'MM/dd/yyyy');  -- Output: '09/22/2023'
```

- ▶ Display ENAME and HIREDATE in MM/DD/YYYY format?

```sql
SELECT ENAME, FORMAT(HIREDATE, 'MM/dd/yyyy') AS HireDateFormatted FROM EMP;
```

- ▶ Waht is DATEADD() function?

```sql
SELECT DATEADD(yy, 1, GETDATE());  -- Adds 1 year
SELECT DATEADD(dd, -10, GETDATE());  -- Subtracts 10 days
```

- ▶ Display yesterday's gold rate?

```sql
SELECT * FROM GOLD_RATES WHERE RATE_DATE = DATEADD(dd, -1, GETDATE());
```

- ▶ Waht is DATEDIFF() function?

```
SELECT DATEDIFF(yy, '2022-09-22', GETDATE());  -- Difference in years
```

- ▶ Display ENAME and experience in years and months?

```
SELECT ENAME,
   DATEDIFF(yy, HIREDATE, GETDATE()) AS YearsExperience,
   DATEDIFF(mm, HIREDATE, GETDATE()) % 12 AS MonthsExperience
FROM EMP;
```

- ▶ Waht is EOMONTH() function?

```
SELECT EOMONTH(GETDATE(), 0);  -- Last day of the current month
```

- ▶ Display the first day of the next month?

```
SELECT DATEADD(dd, 1, EOMONTH(GETDATE()));
```

- ▶ Display the first day of the current year?

```
SELECT DATEFROMPARTS(YEAR(GETDATE()), 1, 1);
```

- ▶ What is String functions in SQL?
- ▶ Waht is UPPER() function?

```
SELECT UPPER('hello');  -- Output: 'HELLO'
```

- ▶ Waht is LOWER() function?

```
SELECT LOWER('HELLO');  -- Output: 'hello'
```

- ▶ Display ENAME and SAL in lowercase:

```sql
SELECT LOWER(ENAME) AS ENAME, SAL FROM EMP;
```

- ▶ Convert names to lowercase in the table:

```sql
UPDATE EMP SET ENAME = LOWER(ENAME);
```

- ▶ Waht is LEN() function?

```sql
SELECT ENAME, LEN(ENAME) AS LEN FROM EMP;
```

- ▶ Display employees' names with 4 characters:

```sql
SELECT * FROM EMP WHERE LEN(ENAME) = 4;
```

- ▶ What is LEFT() function?

```sql
SELECT LEFT('HELLO WELCOME', 5);  -- Output: 'HELLO'
```

- ▶ What is RIGHT() function?

```sql
SELECT RIGHT('HELLO WELCOME', 7);  -- Output: 'WELCOME'
```

- ▶ Employees whose names start with 's':

```sql
SELECT * FROM EMP WHERE LEFT(ENAME, 1) = 's';
```

- ▶ Employees whose names end with 's'

```sql
SELECT * FROM EMP WHERE RIGHT(ENAME, 1) = 's';
```

- ▶ Employees whose names start and end with the same character:

```sql
SELECT * FROM EMP WHERE LEFT(ENAME, 1) = RIGHT(ENAME, 1);
```

- ▶ What is SUBSTRING() function?

```sql
SELECT SUBSTRING('HELLO WELCOME', 7, 4);  -- Output: 'WELC'
```

- ▶ What is REPLICATE() function?

```sql
SELECT REPLICATE('*', 5);  -- Output: '*****'
```

- ▶ Display ENAME and masked SAL:

```sql
SELECT ENAME, REPLICATE('*', LEN(SAL)) AS SAL FROM EMP;
```

- ▶ What is REPLACE() function?

```sql
SELECT REPLACE('hello', 'ell', 'abc');  -- Output: 'habco'
```

- ▶ Employees whose names contain exactly one 'a':

```sql
SELECT * FROM EMP WHERE LEN(ENAME) - LEN(REPLACE(ENAME, 'A', '')) = 1;
```

- ▶ What is TRANSLATE() function?

```sql
SELECT TRANSLATE('hello', 'elo', 'abc');  -- Output: 'habbc'
```

Scenario: Encryption Example:

```sql
SELECT ENAME, TRANSLATE(SAL, '0123456789.', '$bT*h@U%#^&') AS SAL FROM EMP;
```

- ▶ What is STUFF() function?

```
SELECT STUFF('hello welcome', 10, 3, 'abc');  -- Output: 'hello welabce'
```

- ▶ What is CHARINDEX() function?

```
SELECT CHARINDEX('o', 'hello welcome');  -- Output: 5
```

Scenario: Split full names into first and last names:

```
SELECT CID,
   SUBSTRING(CNAME, 1, CHARINDEX(' ', CNAME) - 1) AS FNAME,
   SUBSTRING(CNAME, CHARINDEX(' ', CNAME) + 1, LEN(CNAME)) AS LNAME
FROM CUST;
```

- ▶ Split names into first, middle, and last names:?

```
SELECT CID,
   SUBSTRING(CNAME, 1, CHARINDEX(' ', CNAME) - 1) AS FNAME,
   SUBSTRING(CNAME, CHARINDEX(' ', CNAME) + 1,
             CHARINDEX(' ', CNAME, CHARINDEX(' ', CNAME) + 1) - CHARINDEX('
', CNAME) - 1) AS MNAME,
   SUBSTRING(CNAME, CHARINDEX(' ', CNAME, CHARINDEX(' ', CNAME) + 1) + 1,
LEN(CNAME)) AS LNAME
FROM CUST;
```

- ▶ What is Numeric function in SQL Server?
- ▶ What is ROUND() function?

```
SELECT ROUND(38.5678, 0);  -- Output: 39
SELECT ROUND(38.3647, 0);  -- Output: 38
```

  - Rounding to two decimal places:

```
SELECT ROUND(38.5638, 2);  -- Output: 38.56
SELECT ROUND(38.5678, 2);  -- Output: 38.57
```

  - Rounding to one decimal place:

```
SELECT ROUND(38.5678, 1);  -- Output: 38.6
```

- Rounding to the nearest hundred:

```
SELECT ROUND(386, -2);  -- Output: 400
```

- Rounding to the nearest ten:

```
SELECT ROUND(386, -1);  -- Output: 390
```

- Rounding to the nearest thousand:

```
SELECT ROUND(386, -3);  -- Output: 0
```

- ▶ Round all employee salaries to the nearest hundred

```
UPDATE EMP SET SAL = ROUND(SAL, -2);
```

- ▶ What is the CEILING() function?

```
SELECT CEILING(3.1);  -- Output: 4
```

- ▶ What is the FLOOR() function?

```
SELECT FLOOR(3.9);  -- Output: 3
```

- ▶ What is the POWER() function?

```
SELECT POWER(2, 3);  -- Output: 8  (2 raised to the power of 3)
```

- ▶ What is the ABS() function?

```
SELECT ABS(-5);  -- Output: 5
SELECT ABS(5);   -- Output: 5
```

- ▶ How do these numeric functions help in practical scenarios?
- ▶ What is Conversion Functions?
- ▶ How do you use CAST to convert data types in SQL Server?

```
SELECT CAST(10.5 AS INT); -- Result: 10
```

Example: Display employee names and salaries with string concatenation:

```
SELECT ENAME + ' earns ' + CAST(SAL AS VARCHAR)
FROM EMP;
```

Example: Displaying a message with hire date and job:

```
SELECT ENAME + ' joined on ' + CAST(HIREDATE AS VARCHAR) + ' as ' + JOB FROM
EMP
```

- ▶ How does CONVERT provide more flexibility than CAST?

```
SELECT CONVERT(VARCHAR, GETDATE(), 101); -- Result: '09/26/2023'
```

  - Example: Convert salary to a string with a thousand separator

```
SELECT CONVERT(VARCHAR, SAL, 1) FROM EMP;
```

- ▶ Use CONVERT to display dates in different formats with style numbers.
- ▶ What does CONVERT do with monetary values in SQL Server?
- ▶ How do you use CONVERT to format monetary values?

```
CONVERT(VARCHAR, money, style_number)
```

This converts a monetary value to a string (VARCHAR) using the specified style.

- ▶ What is the default style for monetary conversion with CONVERT?

```
SELECT CONVERT(VARCHAR, SAL, 0) FROM EMP;
```

If SAL is 1234.5, the result will be 1234.50.

- ▶ How do you display monetary values with a thousand separator using CONVERT?

```
SELECT CONVERT(VARCHAR, SAL, 1) FROM EMP;
```

If SAL is 1234.5, the result will be 1,234.50.

- ▶ How can you display employee names and their salaries with a thousand separator using CONVERT?

```
SELECT ENAME, CONVERT(VARCHAR, SAL, 1) AS SAL
FROM EMP;
```

This will show the employee names (ENAME) and their salaries (SAL) formatted with commas as thousand separators.

- ▶ What is the difference between CAST and CONVERT in SQL Server?
- ▶ How does CONVERT provide more flexibility than CAST?

```
SELECT CONVERT(VARCHAR, GETDATE(), 101); -- Result: '09/26/2023'
```

Example: Convert salary to a string with a thousand separator:

```
SELECT CONVERT(VARCHAR, SAL, 1) FROM EMP;
```

- ▶ What is the purpose of the PARSE function in SQL Server?

```
SELECT PARSE('2023-09-26' AS DATETIME); -- Result: 2023-09-26 00:00:00.000
```

- ▶ What does the ISNULL function do in SQL Server?

```
SELECT ISNULL(NULL, 'N/A'); -- Result: 'N/A'
```

- ▶ How does COALESCE differ from ISNULL?

```
SELECT COALESCE(NULL, 'Value1', 'Value2'); -- Result: 'Value1'
```

- ▶ What is the NULLIF function and when would you use it?

```
SELECT ENAME, SAL / NULLIF(COMM, 0) AS SALARY_TO_COMM_RATIO
FROM EMP;
```

- If COMM is 0, NULLIF(COMM, 0) returns NULL, preventing a division by zero error.

- ▶ How can you handle NULL values in SQL Server?
- ▶ What does the ISNULL() function do?

  Example:

```
SELECT ISNULL(100, 200); -- Result: 100
SELECT ISNULL(NULL, 200); -- Result: 200
```

- ▶ How do you use ISNULL() to handle NULL values in calculations?

```
SELECT ENAME, SAL, COMM, SAL + ISNULL(COMM, 0) AS TOTSAL
FROM EMP;
```

- If COMM is NULL, it is replaced with 0 to prevent errors in the calculation.

- ▶ How can you display NULL values as a specific string using ISNULL()?

```
SELECT ENAME, SAL, ISNULL(CAST(COMM AS VARCHAR), 'N/A') AS COMM
FROM EMP;
```

  This displays 'N/A' for NULL commission values.

- ▶ What does the COALESCE() function do?

```
SELECT COALESCE(NULL, NULL, 'default'); -- Result: 'default'
SELECT COALESCE(NULL, 'value1', 'value2'); -- Result: 'value1'
```

- ▶ How do you use COALESCE() to return either salary or commission, or 'N/A'?

```
SELECT ENAME, COALESCE(SAL, COMM, 'N/A') AS INCOME FROM EMP;
```

- This returns the salary if it exists; otherwise, it returns the commission, and if both are NULL, it returns 'N/A'.

- ▶ What is the purpose of the NULLIF() function?

```
SELECT NULLIF(100, 100); -- Result: NULL

SELECT NULLIF(100, 200); -- Result: 100
```

- ▶ How can NULLIF() help avoid division by zero errors?

```
SELECT ENAME, SAL / NULLIF(COMM, 0) AS SALARY_TO_COMM_RATIO FROM EMP;
```

- If COMM is 0, NULLIF(COMM, 0) returns NULL, avoiding a division by zero error.

- ▶ What is the purpose of the RANK() function?

```
SELECT empno, ename, sal,
   RANK() OVER (ORDER BY sal DESC) AS rnk
FROM emp;
```

- ▶ What is the difference between RANK() and DENSE_RANK()?

```
SELECT sal,
   RANK() OVER (ORDER BY sal DESC) AS rnk,
   DENSE_RANK() OVER (ORDER BY sal DESC) AS drnk
FROM emp;
```

- For example, if three employees have the same highest salary, RANK() might assign them all the rank 1 and the next rank will be 4, while DENSE_RANK() will assign all three the rank 1 and the next rank will be 2.

- ▶ How do you use RANK() and DENSE_RANK() to find ranks based on multiple columns?

```
SELECT empno, ename, hiredate, sal,
   DENSE_RANK() OVER (ORDER BY sal DESC, hiredate ASC) AS drnk
FROM emp;
```

- ▶ What does the PARTITION BY clause do with ranking functions?

```
SELECT empno, ename, sal, deptno,
   DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal DESC) AS rnk
FROM emp;
```

- ▶ What is the purpose of the ROW_NUMBER() function?

```
SELECT empno, ename, sal,
   ROW_NUMBER() OVER (ORDER BY empno ASC) AS rno
FROM emp;
```

- ▶ How can LAG() and LEAD() functions be used?

```
SELECT empno, ename, sal,
   LAG(sal, 1) OVER (ORDER BY empno ASC) AS prev_sal
FROM emp;
```

- LEAD() provides access to a row at a given physical offset after the current row.

```
SELECT empno, ename, sal,
   LEAD(sal, 1) OVER (ORDER BY empno ASC) AS next_sal
FROM emp;
```

- ▶ How do you calculate the number of days between hire dates using LAG()?

```
SELECT ename, hiredate,
   DATEDIFF(DD, LAG(hiredate, 1) OVER (ORDER BY hiredate ASC), hiredate) AS
days
FROM emp;
```

- This query calculates the number of days between the current hire date and the previous hire date for each employee.

- ▶ What is the purpus of Aggregate Functions or Group Functions
- ▶ What is the purpose of the MAX() function?

```
SELECT MAX(SAL) FROM EMP; -- Returns the highest salary
```

- ▶ What does the MIN() function do?

```
SELECT MIN(SAL) FROM EMP; -- Returns the lowest salary
```

- ▶ How is the SUM() function used?

```
SELECT SUM(SAL) FROM EMP; -- Returns the total salary

SELECT ROUND(SUM(SAL), -3) FROM EMP; -- Rounds the total salary to the
nearest thousand

SELECT CONVERT(VARCHAR, ROUND(SUM(SAL), -3), 1) FROM EMP; -- Displays the
total salary with a thousand separator

SELECT SUM(SAL + ISNULL(COMM, 0)) FROM EMP; -- Includes commissions in the
total salary
```

- ▶ What does the AVG() function calculate?

```
SELECT AVG(SAL) FROM EMP; -- Returns the average salary

SELECT FLOOR(AVG(SAL)) FROM EMP; -- Rounds down the average salary to the
nearest integer
```

- ▶ How does the COUNT() function differ from COUNT(*)?

```
SELECT COUNT(EMPNO) FROM EMP; -- Counts non-null employee numbers

SELECT COUNT(*) FROM EMP; -- Counts all rows in the table
```

- For a column with values (10, NULL, 20, NULL, 30):
  - COUNT(F1) → Returns 3 (only non-null values)
  - COUNT(*) → Returns 5 (all rows)

- ▶ Is any differences Between COUNT and COUNT(*)?

```

```

-

- ▶ How do you count rows with specific conditions?

```

```

```
SELECT COUNT(*) FROM EMP WHERE HIREDATE LIKE '1981%'; -- Employees who
joined in 1981

SELECT COUNT(*) FROM EMP WHERE DATENAME(DW, HIREDATE) = 'SUNDAY'; --
Employees who joined on a Sunday
```

- ▶ Why can't aggregate functions be used in the WHERE clause?

```
SELECT ENAME FROM EMP WHERE SAL = (SELECT MAX(SAL) FROM EMP); -- Finds
employees with the highest salary
```

- ▶ What is the conditions of finding specific?

```
  SELECT COUNT(*)
  FROM EMP
  WHERE DATENAME(DW, HIREDATE) = 'SUNDAY';
```

- ▶ Is that posible to use aggregate functions in the WHERE clause?

```
SELECT ENAME FROM EMP WHERE SAL = (SELECT MAX(SAL) FROM EMP);
```

- ▶ What other functions can be useful for data manipulation?
- ▶ What is the purpose of the CASE statement in SQL?
    1. Simple CASE
    2. Searched CASE

- ▶ What is a Simple CASE expression?

```
CASE expression
  WHEN value1 THEN result1
  WHEN value2 THEN result2
  ...
  ELSE result
END
```

Example:

```
SELECT ENAME,
   CASE JOB
```

```
        WHEN 'CLERK' THEN 'WORKER'
        WHEN 'MANAGER' THEN 'BOSS'
        WHEN 'PRESIDENT' THEN 'BIG BOSS'
        ELSE 'EMPLOYEE'
    END AS JOB_DESCRIPTION
FROM EMP;
```

- ▶ What is a Searched CASE expression?

  Syntax:

```
CASE
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  ...
  ELSE result
END
```

  Example:

```
SELECT SNO,
    (S1 + S2 + S3) AS TOTAL,
    (S1 + S2 + S3) / 3 AS AVG,
    CASE
        WHEN S1 >= 35 AND S2 >= 35 AND S3 >= 35 THEN 'PASS'
        ELSE 'FAIL'
    END AS RESULT
FROM STUDENT;
```

- ▶ Is that important to use END Keyword?
- ▶ What does the END keyword signify in a CASE statement?
- ▶ How does the CASE statement handle default values?
- ▶ Can you use the CASE statement in an UPDATE query?

```
UPDATE EMP
SET SAL = CASE DEPTNO
          WHEN 10 THEN SAL + (SAL * 0.1)
          WHEN 20 THEN SAL + (SAL * 0.15)
          WHEN 30 THEN SAL + (SAL * 0.2)
          ELSE SAL + (SAL * 0.05)
END;
```

- ▶ Can the CASE statement be used to classify data?

```
SELECT AMT,
   CASE
       WHEN AMT >= 0 THEN 'CREDIT'
       ELSE 'DEBIT'
   END AS TRANS
FROM T1;
```

- ▶ What is the role of the END keyword in a CASE statement?
- ▶ Why is it useful to provide an alias for the CASE statement result?
- ▶ Whay use Aliases?
- ▶ Whay is the importance of aliases?
- ▶ Why is it important to use an alias with the CASE statement?
- ▶ What happens if you don't use an alias with the CASE statement?
- ▶ Can you use the CASE statement in other SQL clauses besides SELECT?
- ▶ What is a practical example of using CASE with an alias in a query?

```
SELECT ENAME,
   CASE JOB
       WHEN 'CLERK' THEN 'WORKER'
       WHEN 'MANAGER' THEN 'BOSS'
       WHEN 'PRESIDENT' THEN 'BIG BOSS'
       ELSE 'EMPLOYEE'
   END AS JOB_DESCRIPTION
FROM EMP;
```

- In this query, JOB_DESCRIPTION is the alias given to the column that shows the job descriptions based on the CASE conditions.

- ▶ What is the purpose of the GROUP BY clause in SQL?
- ▶ How does the GROUP BY clause work with aggregate functions?
- ▶ What is the syntax for using the GROUP BY clause?

```
SELECT columns
FROM tablename
[WHERE condition]
GROUP BY column1, column2, ...
[HAVING condition]
[ORDER BY column ASC/DESC]
```

- ▶ What is the order of execution for SQL queries involving GROUP BY?
- ▶ How does the HAVING clause differ from the WHERE clause?
- ▶ Example: Display department-wise total salary

```
SELECT DEPTNO, SUM(SAL) AS TOTSAL
FROM EMP
GROUP BY DEPTNO;
```

Output:

```
DEPTNO  TOTSAL
10      7000
20      9000
30      2000
```

- ▶ Example: Display job-wise number of employees

```
SELECT JOB, COUNT(*) AS CNT
FROM EMP
GROUP BY JOB;
```

Output:

```
JOB        CNT
ANALYST    2
CLERK      4
MANAGER    3
PRESIDENT  1
SALESMAN   4
```

- ▶ Example: Display year-wise number of employees joined

```
SELECT DATEPART(YY, HIREDATE) AS YEAR, COUNT(*) AS CNT
FROM EMP
GROUP BY DATEPART(YY, HIREDATE);
```

Output:

```
YEAR  CNT
1980  1
1981  10
1982  2
1983  1
```

- ▶ Example: Display day-wise number of employees joined

```
SELECT DATENAME(DW, HIREDATE) AS DAY, COUNT(*) AS CNT
FROM EMP
GROUP BY DATENAME(DW, HIREDATE);
```

- ▶ Example: Display month-wise number of employees joined in 1981

```
SELECT DATENAME(mm, HIREDATE) AS MONTH, COUNT(*) AS CNT
FROM EMP
WHERE DATEPART(YY, HIREDATE) = 1981
GROUP BY DATENAME(mm, HIREDATE);
```

- ▶ Example: Display departments having more than 3 employees

```
SELECT DEPTNO, COUNT(*) AS CNT
  FROM EMP
  GROUP BY DEPTNO
  HAVING COUNT(*) > 3;
```

Output:

```
DEPTNO  CNT
20      5
30      6
```

- ▶ Why do we use the HAVING clause instead of the WHERE clause with aggregate functions?
- ▶ What is the difference bitween WHERE vs. HAVING: Key Differences?
- ▶ Display department-wise number of employees where department numbers are 10 or 20, and the number of employees in each department is greater than 3.

```
SELECT DEPTNO, COUNT(*) AS CNT
  FROM EMP
  WHERE DEPTNO IN (10, 20)
  GROUP BY DEPTNO
  HAVING COUNT(*) > 3;
```

Out:

DEPTNO CNT

DEPTNO	CNT
20	5

Explanation:

- The WHERE clause filters rows where DEPTNO is either 10 or 20.
- The GROUP BY clause groups these rows by DEPTNO.
- The HAVING clause filters these groups to retain only those with more than 3 employees.

- ▶ Find southern states with a population greater than 50 million.

```
SELECT STATE, COUNT(*) AS CNT
FROM PERSONS
WHERE STATE IN ('AP', 'TS', 'KA', 'KL', 'TN')
GROUP BY STATE
HAVING COUNT(*) > 50000000;
```

Out:

STATE	CNT
(example states)	(example states)

Explanation:

- The WHERE clause filters rows for the states 'AP', 'TS', 'KA', 'KL', and 'TN'.
- The GROUP BY clause groups the data by STATE.
- The HAVING clause filters out states with a population count not exceeding 50 million.

- ▶ Whay is Range grouping?
- ▶ Whay is the purpose of GROUPING_ID?
- ▶ What are the Common Values and Their Meanings?
- ▶ Range Grouping: Categorize data into different ranges.

```
SELECT
  CASE
    WHEN SAL BETWEEN 0 AND 2000 THEN '0-2000'
    WHEN SAL BETWEEN 2001 AND 4000 THEN '2001-4000'
    WHEN SAL > 4000 THEN 'ABOVE 4000'
  END AS SALRANGE,
  COUNT(*) AS CNT
FROM EMP
GROUP BY
  CASE
    WHEN SAL BETWEEN 0 AND 2000 THEN '0-2000'
    WHEN SAL BETWEEN 2001 AND 4000 THEN '2001-4000'
    WHEN SAL > 4000 THEN 'ABOVE 4000'
END;
```

Out:

SALRANGE	CNT
0-2000	?
2001-4000	?
ABOVE 4000	?

Explanation:

- CASE Statement: Categorizes salaries into ranges.
- COUNT(*): Counts the number of employees in each salary range.
- GROUP BY: Groups data based on the salary ranges defined in the CASE statement.

- ▶ Display department-wise and within department job-wise total salary.

```
SELECT DEPTNO, JOB, SUM(SAL) AS TOTSAL
FROM EMP
GROUP BY DEPTNO, JOB
ORDER BY DEPTNO ASC;
```

- ▶ Display year-wise and within year quarter-wise number of employees hired.

```
SELECT
  DATEPART(YY, HIREDATE) AS YEAR,
  DATEPART(QQ, HIREDATE) AS QRT,
  COUNT(*) AS CNT
FROM EMP
GROUP BY DATEPART(YY, HIREDATE), DATEPART(QQ, HIREDATE)
ORDER BY YEAR ASC;
```

- Explanation: This query counts the number of employees hired in each quarter of each year, displaying the results sorted by year and quarter.

- ▶ Why do we need GROUPING_ID() with ROLLUP or CUBE?
- For example, when using ROLLUP or CUBE, it might not be clear what values like 32225, 7450, or 14075 represent just by looking at the output. GROUPING_ID() helps identify which rows are subtotals or grand totals by indicating which columns are grouped or aggregated.

- ▶ How does GROUPING_ID() work with ROLLUP and CUBE?
- ▶ Give me the Example of GROUPING_ID() with ROLLUP?

```
SELECT
    Deptno,
    Job,
```

```
    SUM(sal) AS Sumsal,
    GROUPING_ID(Deptno, Job) AS GroupingLevel
FROM Emp
GROUP BY ROLLUP(Deptno, Job)
ORDER BY Deptno, Job;
```

- ▶ What is the practical use of GROUPING_ID() in reports or data analysis?
- ▶ Display state-wise and within state gender-wise population, and also display state-wise and gender-wise subtotals.

```
SELECT
    State,
    Gender,
    SUM(Population) AS TotalPopulation,
    CASE GROUPING_ID(State, Gender)
        WHEN 0 THEN 'Detail'
        WHEN 1 THEN 'State Subtotal'
        WHEN 2 THEN 'Gender Subtotal'
        WHEN 3 THEN 'Grand Total'
    END AS SubtotalType
FROM PopulationTable
GROUP BY CUBE(State, Gender)
ORDER BY State, Gender;
```

Explanation:

- GROUP BY CUBE(State, Gender) generates combinations for all possible groupings: state, gender, state+gender, and the grand total.
- GROUPING_ID(State, Gender) helps identify which combination each row represents:
  - 0 for detailed data (both state and gender are present).
  - 1 for state subtotals.
  - 2 for gender subtotals.
  - 3 for the grand total.

- ▶ Display year-wise and within year quarter-wise total amount, and also display year-wise subtotals.

```
SELECT
    DATEPART(YEAR, TransactionDate) AS Year,
    DATEPART(QUARTER, TransactionDate) AS Quarter,
    SUM(Amount) AS TotalAmount,
    CASE GROUPING_ID(DATEPART(YEAR, TransactionDate), DATEPART    (QUARTER,
TransactionDate))
        WHEN 0 THEN 'Detail'
        WHEN 1 THEN 'Year Subtotal'
        WHEN 2 THEN 'Quarter Subtotal'
        WHEN 3 THEN 'Grand Total'
    END AS SubtotalType
FROM Transactions
```

```
GROUP BY CUBE(DATEPART(YEAR, TransactionDate), DATEPART(QUARTER,
TransactionDate))
ORDER BY Year, Quarter;
```

Explanation:

- DATEPART(YEAR, TransactionDate) extracts the year, and DATEPART(QUARTER, TransactionDate) extracts the quarter from each transaction date.
- GROUP BY CUBE generates all combinations for year and quarter to show the detailed data, subtotals for each year, subtotals for each quarter, and the grand total.
- CASE statement with GROUPING_ID determines the type of row (detail, subtotal, or grand total).

- ▶ Why can't you use column aliases in the GROUP BY clause?

  Example:

```
SELECT DATEPART(YY, HIREDATE) AS YEAR, COUNT(*) AS CNT
FROM EMP
GROUP BY DATEPART(YY, HIREDATE)  /* Valid */
ORDER BY YEAR ASC;               /* Valid */
```

- Here, you must use DATEPART(YY, HIREDATE) in the GROUP BY clause instead of the alias YEAR.

- ▶ Why should non-aggregate columns in the SELECT clause be included in the GROUP BY clause?

  Incorrect Usage:

```
SELECT DEPTNO, ENAME, COUNT(*) AS CNT
FROM EMP
GROUP BY DEPTNO;  /* Invalid */
```

- This will cause an error because ENAME is not included in the GROUP BY clause.

  Correct Usage:

```
SELECT DEPTNO, COUNT(*) AS CNT
FROM EMP
GROUP BY DEPTNO;  /* Valid */
```

  Here, only DEPTNO is included in both SELECT and GROUP BY, and COUNT(*) is an aggregate function.

- ▶ What is the difference between ROLLUP and CUBE in SQL?
- **CUBE:** Generates subtotals for all possible combinations of the specified columns and a grand total. It provides a comprehensive analysis by including subtotals for every possible grouping, regardless of the column order. This can be more processing-intensive but gives a more thorough view of the data.

- ▶ When should I use ROLLUP instead of CUBE?
- ▶ When is it more appropriate to use CUBE?
- ▶ How does ROLLUP handle column combinations differently from CUBE?
- ▶ Why is ROLLUP generally more efficient than CUBE?
- ▶ Can both ROLLUP and CUBE include a grand total?

- Importance of GROUP BY: Groups data by specific columns to perform aggregations like totals, averages, and counts.
- Writing Queries Using GROUP BY: Use GROUP BY to group data and apply aggregate functions.
- WHERE vs HAVING: WHERE filters rows before grouping; HAVING filters groups after aggregation.
- Displaying Subtotals and Grand Total: Use ROLLUP for hierarchical subtotals and CUBE for all possible combinations.
- GROUPING_ID(): Identifies the level of aggregation for each row in a result set, making it clear whether the row is a detail, subtotal, or grand total.

- ▶ What are integrity constraints, and why are they important in databases?
- ▶ What are the different types of integrity constraints?
- **PRIMARY KEY**: A combination of NOT NULL and UNIQUE. Uniquely identifies each record in a table.
  Example: ID INT PRIMARY KEY
  Ensures that ID is unique and not null for every record.
- **CHECK**: Ensures all values in a column satisfy a specific condition.
  Example: Salary DECIMAL(10, 2) CHECK (Salary >= 1000)
  Ensures that Salary must be at least 1000.
- **FOREIGN KEY**: Ensures referential integrity by matching values in one table to another table.
  Example: DeptID INT, FOREIGN KEY (DeptID) REFERENCES Department(ID)
  Ensures that DeptID in one table matches ID in the Department table.
- **DEFAULT**: Provides a default value for a column when no value is specified.
  Example: Status VARCHAR(20) DEFAULT 'Active'
  Automatically sets Status to 'Active' if no value is provided.

- ▶ What is NOT NULL and how to use it?

```
CREATE TABLE emp11 (
  empno INT,
  ename VARCHAR(10) NOT NULL
);

INSERT INTO emp11 VALUES (100, NULL); -- ERROR
INSERT INTO emp11 VALUES (101, 'A');
```

- ▶ What is UNIQUE and how to use it?

```sql
CREATE TABLE cust (
    custid INT,
    cname VARCHAR(10) NOT NULL,
    emailid VARCHAR(20) UNIQUE
);

INSERT INTO cust VALUES (100, 'A', 'abc@gmail.com');
INSERT INTO cust VALUES (101, 'B', 'abc@gmail.com'); -- ERROR
INSERT INTO cust VALUES (102, 'C', NULL);
INSERT INTO cust VALUES (103, 'D', NULL); -- ERROR
```

Note: Depending on the database system, the behavior of UNIQUE with NULLs may vary. Some databases might allow multiple NULLs in a UNIQUE column.

- ▶ What is PRIMARY KEY and how to use it?

```sql
CREATE TABLE emp13 (
    empid INT PRIMARY KEY,
    ename VARCHAR(10) NOT NULL
);

INSERT INTO emp13 VALUES (100, 'A');
INSERT INTO emp13 VALUES (100, 'B'); -- ERROR
INSERT INTO emp13 VALUES (NULL, 'C'); -- ERROR
```

Only one primary key is allowed per table. If you need multiple unique identifiers, declare one column as the primary key and the other columns with the UNIQUE and NOT NULL constraints.

- ▶ What is use of Multiple Unique Columns Example?

```sql
CREATE TABLE cust (
    custid INT PRIMARY KEY,
    cname VARCHAR(10) NOT NULL,
    aadharno NUMERIC(12) UNIQUE NOT NULL,
    panno CHAR(10) UNIQUE NOT NULL
);
```

- ▶ How do the PRIMARY KEY and UNIQUE constraints differ?
- ▶ What is the purpose of the CHECK constraint?
- ▶ Can a table have multiple FOREIGN KEY constraints?
- ▶ How does the DEFAULT constraint work?
- ▶ What is referential integrity, and how is it maintained using the FOREIGN KEY constraint?
- ▶ Can a column have both NOT NULL and DEFAULT constraints?
- ▶ What is the difference between the UNIQUE and PRIMARY KEY constraints?

UNIQUE	PRIMARY KEY
Allows one NULL value (varies by DBMS).	Does not allow NULL values.
Multiple columns in a table can have the UNIQUE constraint.	Only one primary key per table.
Creates a non-clustered index by default.	Creates a clustered index by default (in most DBMS).

- The UNIQUE constraint ensures all values in a column or combination of columns are unique but allows NULL values.
- The PRIMARY KEY constraint uniquely identifies each record and does not allow NULL values.

- ▶ What is a candidate key?

VEHNO	NAME	MODEL	COST	CHASSISNO
1	Car A	X1	10000	123ABC
2	Car B	X2	15000	456DEF

- **Candidate keys:** VEHNO, CHASSISNO
- **Primary key:** VEHNO (selected to uniquely identify records)
- **Secondary key (Alternate key):** CHASSISNO (can also uniquely identify records but is not chosen as the primary key).

- ▶ How are secondary keys declared in a table?

```
CREATE TABLE VEHICLES (
    VEHNO INT PRIMARY KEY,
    CHASSISNO VARCHAR(20) UNIQUE NOT NULL
);
```

- ▶ What is the CHECK constraint, and how does it work?

  syntax:

```
CHECK (condition)
```

  Example:

- Salary must be a minimum of 3000:

```
CREATE TABLE emp15 (
    empid INT PRIMARY KEY,
    ename VARCHAR(10) NOT NULL,
    sal MONEY CHECK(sal >= 3000)
);
```

- Gender must be 'm' or 'f':

```
gender CHAR(1) CHECK (gender IN ('m', 'f'))
```

- Amount must be a multiple of 100:

```
amt MONEY CHECK (amt % 100 = 0)
```

- Password must be a minimum of 6 characters:

```
pwd VARCHAR(10) CHECK (LEN(pwd) >= 6)
```

- Email ID must contain '@' and end with '.com', '.co', or '.in':

```
emailid VARCHAR(20) CHECK (emailid LIKE '%@%' AND
                      (emailid LIKE '%.com' OR
                       emailid LIKE '%.co' OR
                       emailid LIKE '%.in'))
```

- ▶ How can you enforce a password policy that requires lowercase letters, uppercase letters, digits, and special characters (_, @, $)?

```
```sql
CREATE TABLE users (
    userid INT PRIMARY KEY,
    pwd VARCHAR(20) CHECK (
        pwd LIKE '%[a-z]%' AND
        pwd LIKE '%[A-Z]%' AND
        pwd LIKE '%[0-9]%' AND
        (pwd LIKE '%[_]%' OR pwd LIKE '%@%' OR pwd LIKE '%$%')
    )
);
```

This constraint ensures that the password (pwd):
Contains at least one lowercase letter ([a-z]).
Contains at least one uppercase letter ([A-Z]).
Contains at least one digit ([0-9]).
Contains at least one of the special characters (_, @, $).

<hr/>
```

Foran_Key

- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶

- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶

- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶

- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶

- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶

- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶

- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶
- ▶

- ▶
- ▶
- ▶
- ▶
- ▶