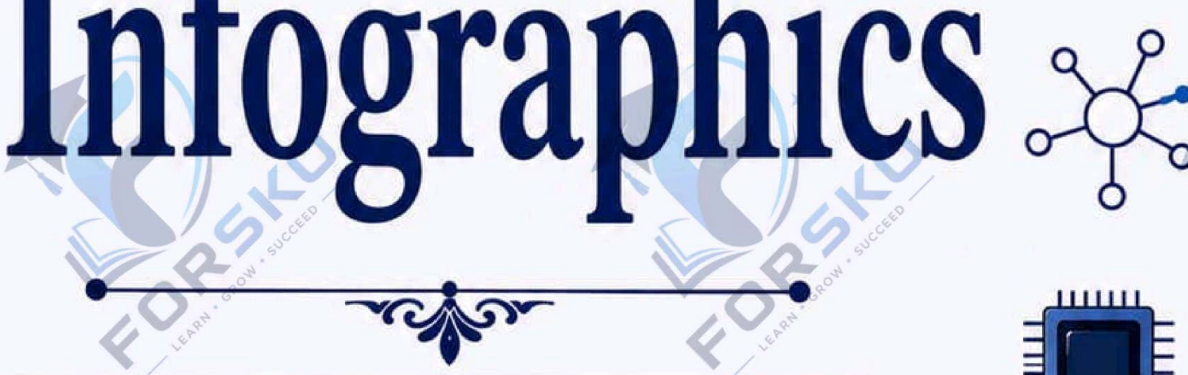
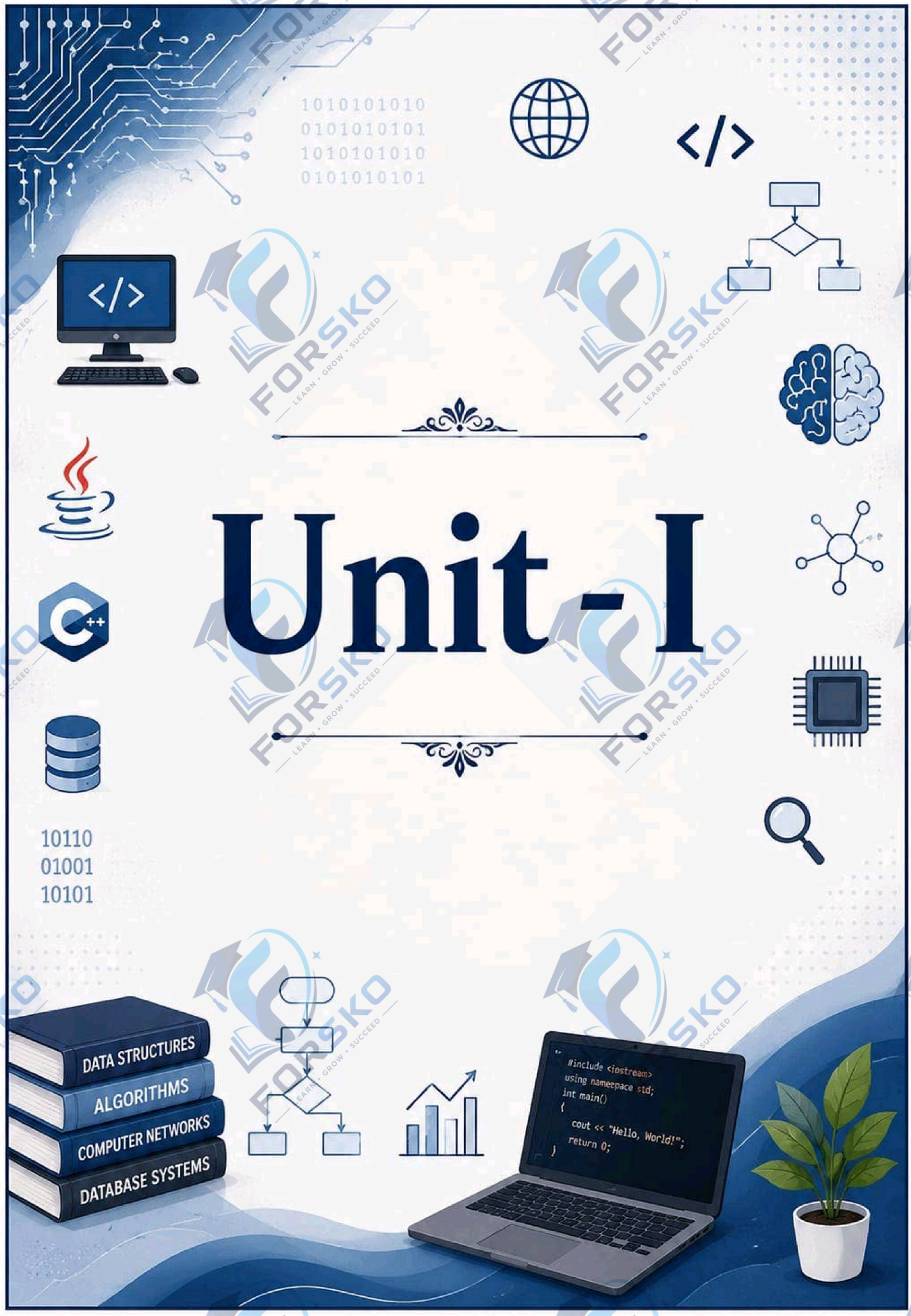
Infographics



Unit-I

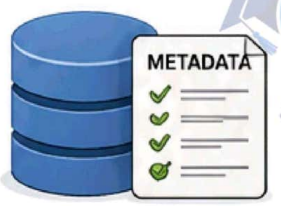

DATA STRUCTURES
ALGORITHMS
COMPUTER NETWORKS
DATABASE SYSTEMS
#include <iostream>
using nameepace std;
int main()
{
cout << "Hello, World!";
return 0;
}

CHARACTERISTICS OF DATABASE

A Database is an organized collection of related data that is stored and managed to meet the needs of multiple users efficiently.

MAIN CHARACTERISTICS

1 SELF-DESCRIBING NATURE

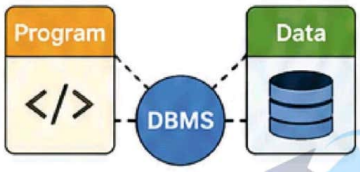


A database contains its own description (metadata) which describes the structure, data types, relationships, and constraints.

2 PROGRAM-DATA INDEPENDENCE

Changes in the data structure do not affect the programs and changes in programs do not affect the data.

3 DATA INDEPENDENCE

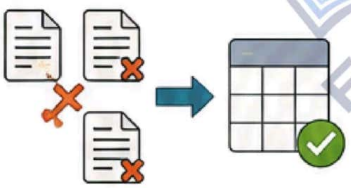


The data can be changed without affecting the application programs that use the data.

4 SHARING OF DATA

Data is shared among multiple users and applications with proper concurrency control.

5 REDUCED DATA REDUNDANCY



Data is stored once and used many times, which reduces duplication and saves storage space.

6 DATA INTEGRITY

Integrity constraints ensure that the data is accurate, consistent and reliable.

7 SECURITY

A database provides security mechanisms to protect data from unauthorized access and misuse.

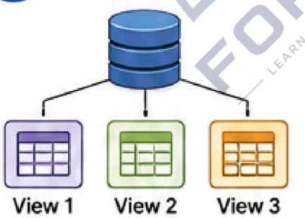
8 CONCURRENCY CONTROL

The database system controls simultaneous access to data by multiple users to avoid conflicts.

9 BACKUP AND RECOVERY

The database system provides backup and recovery facilities to restore data in case of failure.

10 MULTIPLE VIEWS



Different users can have different views of the same data based on their requirements.

SUMMARY

These characteristics make a database system efficient, reliable, secure and easy to use for organizations.

RELATIONAL MODEL

The Relational Model is a way of organizing data in the form of tables (relations) made up of rows and columns. It was proposed by **E.F. Codd in 1970.**

1. WHAT IS RELATIONAL MODEL?

It represents all data in the database as a collection of tables (relations). Each table consists of rows (tuples) and columns (attributes). Relationships between tables are established using keys.

2. BASIC TERMS

- **Relation (Table)** : A table that stores data.
- **Tuple (Row)** : A single row in a table.
- **Attribute (Column)** : A column in a table.
- **Domain** : The set of all possible values for an attribute.
- **Degree** : Number of attributes (columns) in a relation.
- **Cardinality** : Number of tuples (rows) in a relation.

3. STRUCTURE OF A RELATION (TABLE)

STUDENT (A Relation / Table)

RollNo (PK)	Name	Course	Email
101	Arjun	BSc CS	arjun@email.com
102	Neha	BSc IT	neha@email.com
103	Ravi	BSc CS	ravi@email.com
104	Pooja	BSc IT	pooja@email.com

Primary Key (Identifies each row uniquely) — Attributes (Columns) — Tuples (Rows)

4. KEYS IN RELATIONAL MODEL

- **Super Key** : One or more attributes that uniquely identify a tuple.
- **Candidate Key** : Minimal super key (no extra attributes).
- **Primary Key** : One candidate key chosen to identify tuples uniquely.
- **Foreign Key** : Attribute(s) in one table that refer to the primary key of another table.

5. RELATIONSHIP BETWEEN TABLES

STUDENT

RollNo (PK)	Name
101	Arjun
102	Neha
103	Ravi

FK references PK

ENROLLMENT

RollNo (FK)	Course
101	DBMS
102	Python
103	DBMS

ENROLLMENT.RollNo is a Foreign Key (FK) that refers to STUDENT.RollNo which is the Primary Key (PK).

6. FEATURES OF RELATIONAL MODEL

- Data is represented in simple table format.
- Each table has a primary key to uniquely identify rows.
- Relationships are established using keys.
- Reduces data redundancy and inconsistency.
- Supports data integrity and independence.
- Based on mathematical foundation (set theory and logic).

7. EXAMPLE OF RELATIONAL DATABASE

STUDENT

RollNo (PK)	Name	Department
101	Arjun	CS
102	Neha	IT
103	Ravi	CS

1 — N

ENROLLMENT

EnrollID (PK)	RollNo (FK)	CourseID (FK)
1	101	C01
2	102	C02
3	103	C01

N — 1

COURSE

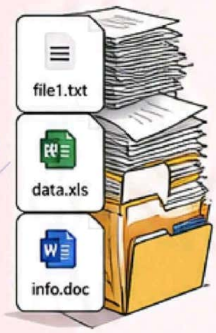
CourseID (PK)	CourseName
C01	DBMS
C02	Python
C03	Data Structures

IN SHORT Relational Model organizes data into related tables using keys and relationships, making data management efficient, consistent and easy to understand.

Need of Database

Why Databases are Essential in Today's World?

Unorganized Files (Traditional Approach)



- ✕ Data is scattered in many files
- ✕ Redundant and duplicate data
- ✕ Inconsistent and unreliable data
- ✕ Difficult to share and access
- ✕ Low security and integrity
- ✕ Hard to maintain and update
- ✕ Time-consuming search

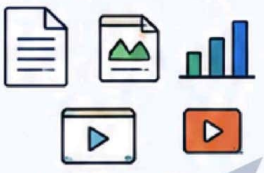
VS

Structured Database (Better Approach)

- ✓ Data is organized in tables
- ✓ Minimizes redundancy
- ✓ Consistent and accurate data
- ✓ Easy data sharing and access
- ✓ High security and integrity
- ✓ Easier maintenance and updates
- ✓ Fast and efficient retrieval

ID	Name	Dept	City
101	Asha	BCA	Pune
102	Rahul	BSA	Delhi
103	Neha	B.Com	Jaipur
...	...	...	...

DATA



Raw facts and figures from various sources

DATABASE MANAGEMENT SYSTEM (DBMS)

Stores, organizes, manages, and processes data efficiently

USEFUL INFORMATION

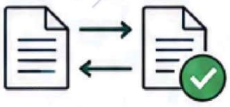
Meaningful, accurate and timely information

10 Key Reasons – Need of Database

1 Organize Large Volumes of Data

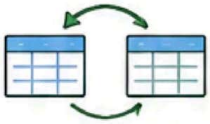
Databases store huge amounts of data in an organized and systematic manner.

2 Reduce Data Redundancy



Eliminates duplicate data and saves storage space and maintenance cost.

3 Improve Data Consistency



Ensures uniform data format and consistent values across the system.

4 Enable Data Sharing

Allows data to be shared across departments, locations and platforms.

5 Provide Data Security

Protects data from unauthorized access, loss, theft or misuse.

6 Support Data Integrity

Enforces accuracy and validity of data using rules and constraints.

7 Allow Fast Data Retrieval

Quickly search, sort and retrieve required data using powerful queries.

8 Offer Backup and Recovery

Regular backup protects data and allows recovery in case of failure.

9 Maintain Data Independence

Changes in data storage or structure do not affect applications or users.

10 Support Multiple Users and Applications

Handles concurrent access by multiple users and different applications.

BSA Semester III – Unit 1: Introduction to Database

SQL and DBMS foundation

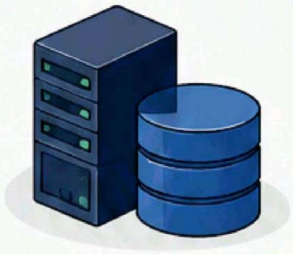
Better Data • Better Decisions • Better Future

ARCHITECTURE OF DBMS

DBMS Architecture defines the structure and components of a Database Management System and how they interact to provide data storage, management and retrieval.

1. Overview

DBMS Architecture shows the various components of a DBMS system and the flow of data between users, queries, and the stored database.



2. Three Level Architecture (ANSI/SPARC)

EXTERNAL LEVEL (VIEW LEVEL)
Multiple user views (external schemas). Each user sees the database as per their requirement.

↔ **External / Conceptual Mapping**
Maps user views to the conceptual level.

CONCEPTUAL LEVEL (LOGICAL LEVEL)
The community view of the database. Describes what data is stored and the relationships among data.

↔ **Conceptual / Internal Mapping**
Maps the conceptual schema to the internal level.

INTERNAL LEVEL (PHYSICAL LEVEL)
Describes how the data is actually stored. Concerned with storage structures and access paths.

↔ **Data Independence**
- Logical Data Independence
- Physical Data Independence

3. Components of DBMS Architecture

Users/Programs
End users or applications interact with the DBMS using queries.

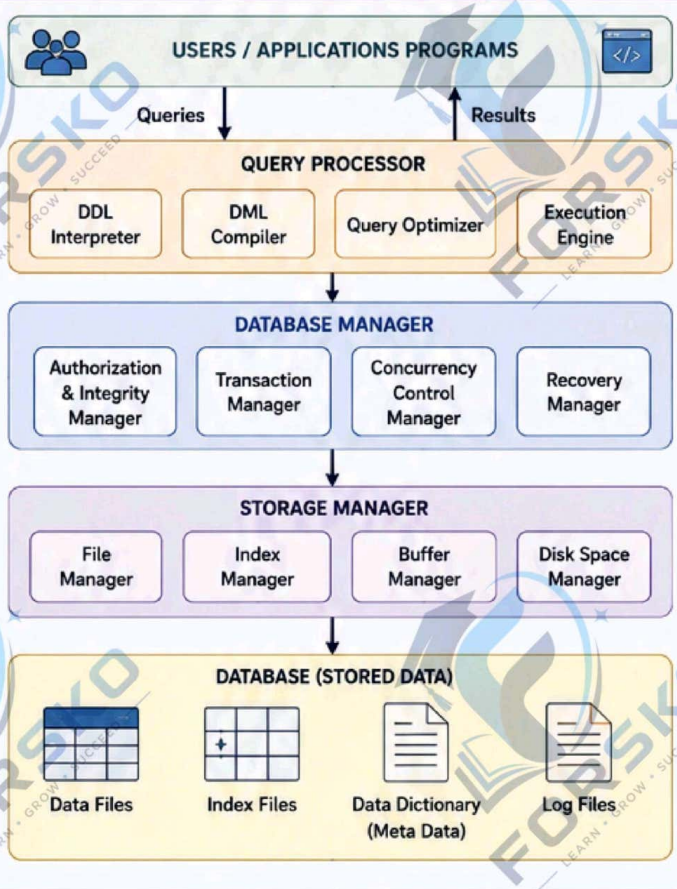
Query Processor
Processes DDL/DML queries, checks syntax and semantics, and generates execution plan.

Database Manager
Coordinates all activities in the system, manages storage structures, transaction and concurrency control.

Storage Manager
Handles the interaction between the low-level data stored in the database and the application programs.

Database
The actual data stored in the form of files, indexes, logs, etc.

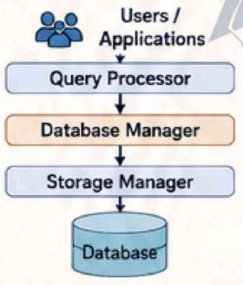
4. Internal Architecture of DBMS



Data Flow

1. Users send queries to DBMS.
2. Query processor interprets and optimizes queries.
3. Database manager manages transactions, security and concurrency.
4. Storage manager accesses data from storage.
5. Results are returned to users.

5. Architecture View



6. Key Points

- Provides data abstraction through three levels.
- Ensures data independence.
- Supports security, integrity and concurrency.
- Optimizes query execution for better performance.
- Manages and organizes data efficiently.

In short: DBMS Architecture ensures that users can access data easily, data is stored securely, and the system works efficiently even with multiple users and large volumes of data.

DBA AND ITS ROLE

A Database Administrator (DBA) is responsible for the overall management, security, performance and availability of the database system to ensure that data is accurate, secure and accessible

1. Who is a DBA?

A DBA is a professional who manages the database environment and ensures that data is stored, organized, secured and available for authorized users.

DBA acts as the guardian of data and the backbone of any organization that relies on databases.

2. Key Responsibilities of a DBA

Database Design
Participates in designing the database structure and defines schema, tables, relationships, constraints.

Installation & Configuration
Installs DBMS Software and configures the system for optimal performance.

Security Management
Creates users, assigns privileges and roles. Ensures data security and access control.

Backup & Recovery
Plans and performs backup regularly and ensures recovery in case of failure or disaster.

Performance Tuning
Monitors and tunes the database for efficient performance and faster query processing.

Storage Management
Manages storage space, data files, indexes and ensures optimal utilization.

Data Integrity
Enforces data accuracy, consistency and integrity using constraints and rules.

Concurrency Control
Manages multiple user access and transactions to ensure data consistency.

3. Skills of a DBA

- Strong knowledge of DBMS (Oracle, MySQL, SQL Server, etc.)
- SQL and query optimization
- Backup and recovery strategies
- Security and user management
- Problem solving and troubleshooting
- Performance monitoring and tuning
- Communication and documentation

4. DBA Role in the Organization

Ensures Data Availability
Keeps the database up and running for users.

Ensures Data Security
Protects data from unauthorized access, loss or corruption.

Improves Performance
Optimizes database performance for faster access and processing.

Supports Business Decisions
Provides reliable and accurate data for analysis and reporting.

Ensures Business Continuity
Plans for backup and recovery to avoid data loss and downtime.

5. DBA Works With

End Users	Developers	System Administrators	Management	Auditors
To understand data requirements and provide access.	To design database objects and improve application performance.	For hardware, OS and network related configurations.	To support reporting, analytics and decision making.	To ensure compliance with standards and regulations.

6. Types of DBA

- **System DBA** – Handles overall database system and infrastructure.
- **Application DBA** – Works with application databases and supports developers.
- **Security DBA** – Focuses on database security and user access control.
- **Data Warehouse DBA** – Manages data warehouses and large volumes of data.

7. Importance of DBA

Protects valuable data assets	Ensures high performance and efficiency	Minimizes data loss and system downtime	Supports smooth operations and business growth	Ensures compliance with policies and regulations

> A good DBA ensures that the right data is available to the right people at the right time in the right way.

In Short: DBA plans, manages, secures and optimizes the database system so that organizations can store data safely and use it effectively.

STRUCTURED QUERY LANGUAGE (SQL) INTRODUCTION

SELECT *
FROM Students;

SQL (Structured Query Language) is a standard programming language used to **create, manage, manipulate and retrieve data** from relational databases.

1. What is SQL?

- SQL is a non-procedural (declarative) language.
- It is used to communicate with databases.
- SQL follows a simple English-like syntax, making it easy to learn and use.
- SQL is an ANSI (American National Standards Institute) standard language.

2. Purpose of SQL

- Create and modify database structures.
- Insert, update, delete and retrieve data.
- Control access to data.
- Maintain data integrity and security.
- Perform transactions and manage concurrency.

3. Key Features of SQL

Standard Language
SQL is a standardized language for database systems.

Easy to Learn
Uses simple, English-like commands.

Powerful and Flexible
Supports complex queries and data operations.

Data Independence
Separates how data is stored from how it is used.

Security
Provides built-in security and access control.

Set-Based Language
Works on sets of data, not individual records.

4. SQL Works with Relational Databases

SQL works with data stored in tables (relations) which are made up of rows and columns.

Example Table: Students

ID	Name	Course	City
101	Asha	BCA	Pune
102	Rahul	BSA	Delhi
103	Neha	B.Com	Jaipur

Column (Attribute)

Row (Tuple)

SQL helps us perform operations on this data such as retrieving, inserting, updating and deleting.

5. Why is SQL Important?

It is widely used in various applications like banking, e-commerce, education, healthcare, etc.

It is supported by almost all database systems (MySQL, Oracle, SQL Server, PostgreSQL, etc.).

It is essential for data analysts, developers, DBAs and IT professionals.

6. Example: Simple SQL Queries

Query	Description
SELECT * FROM Students;	Retrieves all records from Students table.
SELECT Name, City FROM Students;	Retrieves Name and City columns.
SELECT * FROM Students WHERE City = 'Pune';	Retrieves students from Pune.
INSERT INTO Students (ID, Name, Course, City) VALUES (104, 'Karan', 'BCA', 'Mumbai');	Inserts a new record.
UPDATE Students SET City = 'Hyderabad' WHERE ID = 101;	Updates the city of student with ID 101.
DELETE FROM Students WHERE ID = 103;	Deletes the record with ID 103.

7. SQL at a Glance

User
User writes SQL Query

→

SQL Query
Query is sent to the database

→

Database Engine
Processes the query and accesses data

→

Result Set
Data is retrieved as result

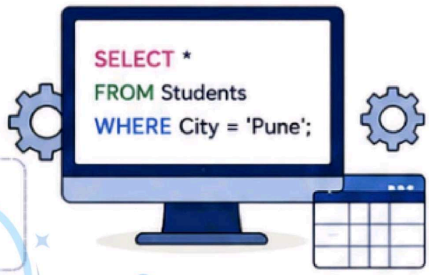
→

Output
Results are displayed to the user

★ In Short

SQL is the foundation for managing and interacting with relational databases. It allows us to store, access, manipulate and secure data efficiently.

SQL FEATURES & CHARACTERISTICS



SQL (Structured Query Language) is a powerful, standard and user-friendly language used to create, access and manage data in relational databases.

SQL FEATURES

1 Standard Language
SQL is an ANSI/ISO standard language (ISO/IEC 9075). It is widely accepted and used across different DBMS.

2 Easy to Learn and Use
SQL has simple English-like syntax and keywords, making it easy to learn and understand.

3 Powerful and Flexible
Supports a wide range of operations such as data retrieval, insertion, update, delete, and complex queries.

4 Data Manipulation
Allows users to insert, update, delete and retrieve data efficiently from one or more tables.

5 Data Security
Provides access control, user authentication and privileges to protect data from unauthorized access.

6 Support for Relationships
Easily handles relationships between tables using keys (Primary Key, Foreign Key) and joins.

7 High Performance
Optimized query processing and indexing improve the speed and efficiency of data operations.

8 Scalability
SQL is suitable for small applications as well as large enterprise-level database systems.

SQL CHARACTERISTICS

1 Declarative (Non-Procedural)
SQL is declarative, not procedural. We tell the system what data we want, not how to get it.

2 Set-Oriented Language
SQL works on sets of data (rows), not on individual records.

3 Used for Relational Databases
SQL is specifically designed for relational database management systems (RDBMS).

4 High-Level Language
SQL is a high-level language. It hides the internal storage details from the user.

5 Data Independence
Provides both Logical Data Independence and Physical Data Independence.

6 Portable
SQL queries are portable across different database systems with little or no modification.

7 Integrity Constraints
SQL supports constraints like PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, NOT NULL to ensure data accuracy.

8 Transaction Management
SQL supports ACID properties (Atomicity, Consistency, Isolation, Durability) to ensure reliable transactions.

9 Multi-User Support
SQL allows multiple users to access and work on the database simultaneously.

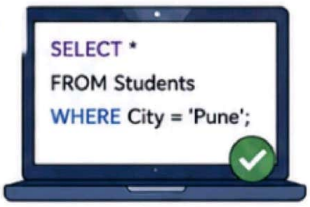
10 Extensible
SQL can be extended with additional features, functions and procedures to meet application needs.

In Short

SQL is a standard, powerful, flexible and easy-to-use language with strong features and characteristics that make it the ideal choice for managing relational databases efficiently and securely.

SQL ADVANTAGES

SQL (Structured Query Language) provides an efficient and standardized way to manage and interact with data in relational database systems.



1 Easy to Learn and Use

SQL has a simple English-like syntax that is easy to learn for beginners and powerful enough for advanced users.

2 Standard Language

SQL is an ANSI/ISO standard (ISO/IEC 9075) used by almost all relational database systems, ensuring compatibility.

3 Works with Relational Databases

Designed specifically for relational databases, SQL efficiently manages data stored in tables and the relationships between them.

4 High Performance

SQL queries are optimized by the database engine for fast data retrieval and manipulation, improving application performance.

5 Supports Multiple Users

Allows multiple users to access and work on data simultaneously while maintaining data integrity.

6 Data Integrity and Security

SQL provides strong security features and supports constraints (primary key, foreign key, unique, check, not null) to ensure accurate and consistent data.

7 Data Independence

Changes in the database structure do not affect the application programs (logical and physical data independence is supported).

8 Flexibility

SQL supports a wide range of operations (data definition, manipulation, control) and can handle simple to complex queries easily.

9 Scalability

SQL is suitable for small applications as well as large enterprise-level database systems.

10 Integration

SQL can be easily integrated with various programming languages, tools and applications.

SQL DATA TYPES

SQL data types define the type of data that can be stored in a column of a table.

Category	Data Types	Description	Example
Numeric Types	INT / INTEGER, SMALLINT, BIGINT DECIMAL(p,s) / NUMERIC(p,s) FLOAT(p), REAL, DOUBLE PRECISION	Store whole numbers, decimal numbers and approximate floating-point numbers.	101, -50, 1000 12.50, 999.99 3.14, 2.71828
Character/ String Types	CHAR(n), VARCHAR(n), TEXT / CLOB	Store fixed-length, variable-length or large text data.	'A', 'SQL' 'John Doe', 'Pune' 'This is a long text....'
Date & Time Types	DATE, TIME, TIMESTAMP, DATETIME, YEAR	Store date, time or date-time values.	'2024-05-20' '14:30:00' '2024-05-20 14:30:00'
Boolean Type	BOOLEAN	Store true/false values.	TRUE, FALSE
Binary Types	BINARY(n), VARBINARY(n), BLOB	Store binary data such as images, audio, files.	X'ABCD' (Binary data)
Other Types	ENUM, SET, JSON, XML	Store predefined values, multiple values, JSON or XML data.	ENUM('Male','Female') JSON object XML document

Key Points

- Choosing the right data type improves storage efficiency and performance.
- Some data types may vary slightly across different DBMS (MySQL, Oracle, SQL Server, PostgreSQL, etc.).
- Always use appropriate size (n, p, s) to manage memory effectively.

SQL provides a rich set of data types to handle different kinds of data accurately and efficiently.

SQL DATA TYPES & OPERATORS

SQL (Structured Query Language) is used to create, manage and retrieve data from relational databases.

```
SELECT *
FROM Students
WHERE City = 'Pune';
```

SQL DATA TYPES

SQL data types define the kind of data that can be stored in a column of a table.

Category	Data Types	Description	Example	Storage / Range
1. Numeric Types	INT / INTEGER SMALLINT BIGINT DECIMAL(p,s) NUMERIC(p,s) FLOAT(p) REAL DOUBLE PRECISION	Store whole numbers, decimal numbers and approximate floating-point numbers.	101 -50 12345 12.50 3.1415	INT: -2,147,483,648 to 2,147,483,647 DECIMAL(5,2): -999.99 to 999.99 FLOAT: Approximate numeric values
2. Character / String Types	CHAR(n) VARCHAR(n) TEXT CLOB	Store fixed-length or variable-length character data.	'A' 'John Doe' 'Hello World'	CHAR(n): Fixed length VARCHAR(n): 0 to n characters TEXT/CLOB: Large text data
3. Date & Time Types	DATE TIME TIMESTAMP DATETIME YEAR	Store date, time or date-time values.	'2024-05-20' '14:30:00' '2024-05-20 14:30:00' '2024-05-20 14:30:00' '2024'	DATE: YYYY-MM-DD TIME: HH:MI:SS TIMESTAMP: Date + Time YEAR: YYYY
4. Boolean Type	BOOLEAN	Store true/false values.	TRUE FALSE	TRUE or FALSE (1 or 0 in some DBMS)
5. Binary Types	BINARY(n) VARBINARY(n) BLOB	Store binary data such as images, audio, files.	X'1A2B' (Binary data)	Up to n bytes or large binary objects (BLOB)
6. Other / Advanced Types	ENUM SET JSON XML	Store predefined values, multiple values, JSON or XML data.	ENUM('Male', 'Female') SET('A','B','C') {"name":"Asha"} <note>Data</note>	DBMS specific support (MySQL, PostgreSQL, Oracle, SQL Server, etc.)

Note: The exact size and range may vary slightly between different DBMS (MySQL, Oracle, SQL Server, PostgreSQL, etc.). Always use appropriate data types to save storage and ensure data integrity.

SQL OPERATORS

Operators are special symbols or keywords used to perform operations on data and values in SQL.

1. Arithmetic Operators

Operator	Meaning	Example	Result
+	Addition	10 + 5	15
-	Subtraction	10 - 5	5
*	Multiplication	10 * 5	50
/	Division	10 / 5	2
%	Modulus (Remainder)	10 % 3	1

2. Comparison Operators

Operator	Meaning	Example	Result
=	Equal to	A = B	True/False
<> or !=	Not equal to	A <> B	True/False
>	Greater than	A > B	True/False
<	Less than	A < B	True/False
>=	Greater than or equal to	A >= B	True/False
<=	Less than or equal to	A <= B	True/False

3. Logical Operators

Operator	Meaning	Example
AND	True if both conditions are true	A > 10 AND B < 20
OR	True if any one condition is true	A > 10 OR B < 20
NOT	Negates the condition	NOT (A > 10)

4. Pattern Matching Operators

Operator	Meaning	Example	Matches
LIKE	Matches a pattern	'A%'	Starts with 'A'
_ (underscore)	Matches any single character	'A_B'	A followed by any one char then B
% (percent)	Matches any sequence of chars	'%xyz%'	Contains 'xyz' anywhere

5. IN / Between / IS Operators

Operator	Meaning	Example
IN	Matches any value in a list	City IN ('Pune', 'Delhi')
NOT IN	Not matches in a list	City NOT IN ('Pune')
BETWEEN	Between a range	Age BETWEEN 18 AND 25
NOT BETWEEN	Not between a range	Age NOT BETWEEN 18 AND 25
IS NULL	Checks for NULL	Address IS NULL
IS NOT NULL	Checks for NOT NULL	Address IS NOT NULL

6. Bitwise Operators

Operator	Meaning	Example
&	Bitwise AND	A & B
|	Bitwise OR	A | B
^	Bitwise XOR	A ^ B
~	Bitwise NOT	~A
<<	Left shift	A << 2
>>	Right shift	A >> 2

7. Assignment Operators

Operator	Meaning	
=	Assign	@x = 10
+=	Add and assign	@x += 5 (Equivalent to @x = @x + 5)
-=	Subtract and assign	@x -= 5 (Equivalent to @x = @x - 5)
*=	Multiply and assign	@x *= 5 (Equivalent to @x = @x * 5)
/=	Divide and assign	@x /= 5 (Equivalent to @x = @x / 5)
%=	Modulus and assign	@x %= 5 (Equivalent to @x = @x % 5)

8. Other Operators

Operator	Meaning	Example
+ (Concatenation)	Concatenate strings	'John' + ' Doe'
()	Group expressions	(A + B) * C
,	Separator (in lists)	SELECT id, name FROM Students
EXISTS	Checks existence of rows	WHERE EXISTS (SELECT 1 FROM Orders)
ANY / ALL	Compare with any or all values	price > ANY (SELECT price FROM Products)

In Short

SQL Data Types define what kind of data we can store.
SQL Operators help us perform calculations, comparisons, and logical operations on that data.

COMPONENTS OF SQL

SQL is divided into different components (sub-languages) based on the type of operations they perform.

```
SELECT *
FROM Students
WHERE City = 'Pune';
```

1. DDL (Data Definition Language)

Used to define and manage the structure of database objects.

Example Commands: CREATE, ALTER, DROP, TRUNCATE, RENAME

2. DML (Data Manipulation Language)

Used to retrieve and manipulate the data in database objects.

Example Commands: SELECT, INSERT, UPDATE, DELETE

3. DCL (Data Control Language)

Used to control access and permissions on data in the database.

Example Commands: GRANT, REVOKE

4. TCL (Transaction Control Language)

Used to manage transactions in the database.

Example Commands: COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION

DDL (Data Definition Language)

DDL commands are used to define, modify and manage the database structure (schemas, tables, indexes, constraints, etc.). These changes affect the structure, not the data.

Common DDL Commands

Command	Purpose	Example	Description
CREATE	Creates a new database object (database, table, view, index, etc.).	`CREATE TABLE Students ( ID INT PRIMARY KEY, Name VARCHAR(50), City VARCHAR(30) );`	Creates a new table named 'Students'.
ALTER	Modifies the structure of an existing database object.	`ALTER TABLE Students ADD Email VARCHAR(50);`	Adds a new column 'Email' in existing table 'Students'.
DROP	Deletes an existing database object completely.	`DROP TABLE Students;`	Deletes the table 'Students' and its structure.
TRUNCATE	Removes all records from a table, but keeps the structure intact.	`TRUNCATE TABLE Students;`	Deletes all rows from the table 'Students' but table structure remains.
RENAME	Renames an existing database object.	`RENAME TABLE Students TO Student_Info;`	Renames table 'Students' to 'Student_Info'.

Key Points

- DDL commands work on the structure (schema) of the database.
- Changes made by DDL commands are auto committed.
- DDL defines how data is stored and organized in the database.
- Only authorized users can execute DDL commands.
- Examples of objects managed by DDL: Tables, Views, Indexes, Schemas, Constraints.

Example: DDL Workflow

1. **CREATE** — `CREATE TABLE Students (....);`
2. **ALTER** — `ALTER TABLE Students ADD Email VARCHAR(50);`
3. **DROP** — `DROP TABLE Students;`
4. **TRUNCATE** — `TRUNCATE TABLE Students;`
5. **RENAME** — `RENAME TABLE Students TO Student_Info;`

In Short

DDL commands are the foundation of any database.
They define the structure that stores and organizes the data.

COMPONENTS OF SQL – DML
(DATA MANIPULATION LANGUAGE)

```
SELECT *
FROM Students
WHERE City = 'Pune';
```

DML commands are used to retrieve, insert, update and delete data in database tables. These commands work with the data stored in the database.

DML COMPONENTS

1 SELECT

Retrieves data from one or more tables.

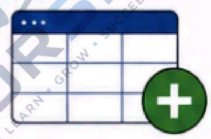
Syntax

```
SELECT column1, column2, ...
FROM table_name
[WHERE condition]
[ORDER BY column]
[GROUP BY column]
[HAVING condition];
```

Example

```
SELECT Name, City
FROM Students
WHERE City = 'Pune';
```

2 INSERT



Inserts new records into a table.

Syntax

```
INSERT INTO table_name
(column1, column2, ...)
VALUES (value1, value2, ...);
```

Example

```
INSERT INTO Students
(ID, Name, City, Course)
VALUES (101, 'Asha', 'Pune',
'BCA');
```

3 UPDATE

Modifies existing records in a table.

Syntax

```
UPDATE table_name
SET column1 = value1,
    column2 = value2, ...
WHERE condition;
```

Example

```
UPDATE Students
SET City = 'Mumbai'
WHERE ID = 101;
```

4 DELETE

Deletes existing records from a table.

Syntax

```
DELETE FROM table_name
WHERE condition;
```

Example

```
DELETE FROM Students
WHERE ID = 101;
```

EXAMPLE TABLE: Students

ID	Name	City	Course	Age
101	Asha	Pune	BCA	20
102	Rahul	Delhi	BSA	21
103	Neha	Mumbai	B.Com	19

DML IN ACTION

- **SELECT** – Retrieves data → View or Report
- **INSERT** – Adds new data → New record created
- **UPDATE** – Modifies data → Existing record updated
- **DELETE** – Removes data → Record deleted

KEY POINTS

- DML works on the data (rows) of tables.
- It does not change the structure of the table.
- Changes made by DML commands are temporary until saved using TCL (COMMIT / ROLLBACK).
- Proper WHERE clause is important in UPDATE and DELETE to avoid affecting all rows.

DML VS DDL

Aspect	DML	DDL
Purpose	Manipulate data	Define structure
Affects	Data (Rows)	Structure (Tables, Schema)
Examples	SELECT, INSERT, UPDATE, DELETE	CREATE, ALTER, DROP, TRUNCATE
Auto Commit	No (Requires COMMIT)	Yes (Auto Commit)

In Short

DML is the heart of data handling in SQL. It allows users to retrieve, add, modify and remove data efficiently from database tables.

COMPONENTS OF SQL – DCL
(DATA CONTROL LANGUAGE)

```
SELECT *
FROM Students
WHERE City = 'Pune';
```

DCL commands are used to control access and permissions on database objects. They help keep data secure by allowing only authorized users to access or perform specific actions.

DCL COMPONENTS

SQL has two main DCL commands.

1 GRANT

GRANT is used to give privileges (permissions) to users or roles on database objects.

Syntax

```
GRANT privilege_list
 ON object_name
 TO user_name [, user_name] ...
 [WITH GRANT OPTION];
```

Example

```
GRANT SELECT, INSERT ON Students
TO user1;

GRANT ALL PRIVILEGES ON *.*
TO user2 WITH GRANT OPTION;
```

Common Privileges

SELECT	INSERT	UPDATE	DELETE	EXECUTE
Read data	Add data	Modify data	Remove data	Run procedures

2 REVOKE

REVOKE is used to remove privileges that were previously granted to users or roles.

Syntax

```
REVOKE privilege_list
 ON object_name
 FROM user_name [, user_name] ...
 [CASCADE | RESTRICT];
```

Example

```
REVOKE INSERT ON Students
FROM user1;

REVOKE ALL PRIVILEGES ON *.*
FROM user2 CASCADE;
```

Notes

- CASCADE – Removes the privilege and all rights granted by that user to others.
- RESTRICT – Does not allow revoke if the privilege has been granted to other users.

OBJECTS ON WHICH PRIVILEGES CAN BE APPLIED

Tables | Views | Databases | Stored Procedures | Functions | Sequences

EXAMPLE: DCL IN ACTION

ADMIN

1. **GRANT** – Admin grants SELECT privilege on Students table to user1.

```
GRANT SELECT ON Students
TO user1;
```

2. **USER1 ACCESS** – user1 can now view data from Students table.

```
SELECT *
FROM Students;
```

3. **REVOKE** – Admin revokes SELECT privilege from user1.

```
REVOKE SELECT ON Students
FROM user1;
```

USER1

4. **ACCESS REMOVED** – user1 can no longer view data from Students table.

```
SELECT * FROM Students;
-- Error: You do not have
   permission to execute this
   operation.
```

KEY POINTS

- DCL is used for security and access control.
- GRANT gives privileges to users.
- REVOKE removes privileges from users.
- Privileges can be object specific (e.g., a table) or global.
- Only users with appropriate privileges (e.g., ADMIN) can grant or revoke access.
- Helps enforce data security and maintain integrity.

DCL SUMMARY

Command	Purpose	Example	Effect
GRANT	Gives privileges to users	GRANT SELECT ON Students TO user1;	user1 can access the table
REVOKE	Removes privileges from users	REVOKE SELECT ON Students FROM user1;	user1 can no longer access the table

In Short

DCL commands (GRANT, REVOKE) are essential for controlling who can access what in a database. They ensure that only authorized users can perform actions on database objects, keeping your data safe and secure.

COMPONENTS OF SQL - TCL
(TRANSACTION CONTROL LANGUAGE)

```
SELECT *
FROM Students
WHERE City = 'Pune';
```

TCL commands are used to manage transactions in a database. They ensure data integrity by allowing you to save, undo, or manage a group of SQL operations as a single unit of work.

TCL COMPONENTS

SQL has four main TCL commands.

1 COMMIT

Saves all changes made in the current transaction permanently.

Syntax

```
COMMIT;
```

Example

```
UPDATE Students
SET City = 'Pune'
WHERE ID = 101;
COMMIT;
```

2 ROLLBACK

Undoes all changes made in the current transaction and restores the database to the last committed state.

Syntax

```
ROLLBACK;
```

Example

```
UPDATE Students
SET City = 'Pune'
WHERE ID = 101;
ROLLBACK;  -- Changes undone
```

3 SAVEPOINT

Sets a point within a transaction to which you can later ROLLBACK.

Syntax

```
SAVEPOINT savepoint_name;
```

Example

```
SAVEPOINT sp1;
UPDATE Students SET City = 'Pune'
WHERE ID = 101;
ROLLBACK TO sp1;
```

4 ROLLBACK TO SAVEPOINT

Rolls back the transaction to a specific SAVEPOINT, without affecting earlier changes.

Syntax

```
ROLLBACK TO savepoint_name;
```

Example

```
SAVEPOINT sp1;
UPDATE Students SET City = 'Pune'
WHERE ID = 101;
ROLLBACK TO sp1;
```

HOW TRANSACTIONS WORK

1. Start Transaction
A series of SQL statements are executed.

→

2. Perform Operations
INSERT / UPDATE / DELETE operations are performed.

→

3. Decide
Choose to COMMIT or ROLLBACK.

→

4A. COMMIT
Changes are saved permanently.

OR

4B. ROLLBACK
Changes are discarded (not saved).

EXAMPLE SCENARIO

Transfer ₹500 from Account A to Account B.
(Two update operations must succeed together.)

Step	Action	Description	Account A	Account B
1	Initial State	Before transaction	₹1000	₹500
2	UPDATE A	Debit ₹500 from A	₹500	₹500
3	UPDATE B	Credit ₹500 to B	₹500	₹1000
4A	COMMIT	All changes saved	₹500	₹1000
4B	ROLLBACK	All changes undone	₹1000	₹500

Using TCL ensures either both operations succeed (COMMIT) or both are undone (ROLLBACK), maintaining data integrity.

KEY POINTS

- A transaction is a group of one or more SQL operations treated as a single unit of work.
- COMMIT makes changes permanent.
- ROLLBACK undoes all changes since the last COMMIT.
- SAVEPOINT allows partial rollback within a transaction.
- TCL commands help maintain ACID properties (Atomicity & Consistency) in a database.
- By default, most databases use Auto-Commit Mode (each statement is committed automatically).

TCL COMMANDS SUMMARY

Command	Purpose	Scope	Effect	Example
COMMIT	Save all changes in the transaction	Current Transaction	Makes all changes permanent	`COMMIT;`
ROLLBACK	Undo all changes in the transaction	Current Transaction	Discards all uncommitted changes	`ROLLBACK;`
SAVEPOINT	Set a point within a transaction	Current Transaction	Marks a point to which you can rollback	`SAVEPOINT sp1;`
ROLLBACK TO SAVEPOINT	Undo changes back to a savepoint	From Savepoint to Current	Keeps changes before the savepoint intact	`ROLLBACK TO sp1;`

IN SHORT

TCL commands (COMMIT, ROLLBACK, SAVEPOINT, ROLLBACK TO SAVEPOINT) manage transactions to ensure reliable and consistent data.

SQL SELECT STATEMENT
with Clauses

```
SELECT *
FROM Students
WHERE City = 'Pune';
```

The SELECT statement is used to retrieve data from one or more tables.
Clauses help filter, group, and sort the data as needed.

1 WHERE

Filters rows based on a specified condition.

Syntax

```
SELECT column_list
FROM table_name
WHERE condition;
```

Example

```
SELECT Name, City
FROM Students
WHERE City = 'Pune';
```

2 GROUP BY

Groups rows that have the same values in specified column(s).

Syntax

```
SELECT column_list
FROM table_name
GROUP BY column_name;
```

Example

```
SELECT City, COUNT(*) AS Total
FROM Students
GROUP BY City;
```

3 HAVING

Filters groups based on a condition.

Syntax

```
SELECT column_list
FROM table_name
GROUP BY column_name
HAVING condition;
```

Example

```
SELECT City, COUNT(*) AS Total
FROM Students
GROUP BY City
HAVING COUNT(*) > 1;
```

4 ORDER BY

Sorts the result set in ascending (ASC) or descending (DESC) order.

Syntax

```
SELECT column_list
FROM table_name
ORDER BY column_name [ASC | DESC];
```

Example

```
SELECT Name, Marks
FROM Students
ORDER BY Marks DESC;
```

SAMPLE TABLE: Students

ID	Name	City	Course	Marks
1	Asha	Pune	BCA	85
2	Rahul	Delhi	BCA	78
3	Neha	Pune	BBA	92
4	Rohan	Mumbai	BCA	87
5	Priya	Pune	BCA	90
6	Karan	Delhi	BBA	76
7	Sneha	Mumbai	BBA	88
8	Mohit	Pune	BCA	65

USING ALL CLAUSES TOGETHER

You can combine clauses in the following logical order:

WHERE → GROUP BY → HAVING → ORDER BY

```
SELECT City, COUNT(*) AS Total_Students, AVG(Marks) AS Avg_Marks
FROM Students
WHERE Course = 'BCA'
GROUP BY City
HAVING COUNT(*) > 1
ORDER BY Avg_Marks DESC;
```

Result:

City	Total_Students	Avg_Marks
Pune	3	80.00
Mumbai	1	87.00

Explanation:

- Get students of BCA course,
  group by City,
  keep groups having more than 1 student, and
  sort by average marks in descending order.

ORDER OF CLAUSE APPLICATION

1. **WHERE** — Filters individual rows.
2. **GROUP BY** — Groups the filtered rows.
3. **HAVING** — Filters the groups.
4. **ORDER BY** — Sorts the final result.

→ Final Result

QUICK REFERENCE

Clause	Purpose	Can Use Aggregate Functions?	Filters
WHERE	Filters rows before grouping	No	Row level
GROUP BY	Groups rows with same values	Yes	Groups rows
HAVING	Filters groups after grouping	Yes	Group level
ORDER BY	Sorts the final output	No	Final result

KEY POINTS

- WHERE filters rows before grouping.
- GROUP BY groups rows.
- HAVING filters groups after grouping.
- ORDER BY sorts the final result set.
- The order of clauses (WHERE → GROUP BY → HAVING → ORDER BY) is important and must be followed.
- Aggregate functions: COUNT(), SUM(), AVG(), MIN(), MAX() are used with GROUP BY and HAVING.

In Short

SELECT + clauses help you retrieve the right data from your database.
WHERE filters rows, GROUP BY groups them, HAVING filters groups,
and ORDER BY sorts the final result.

DATA AND INFORMATION

From Raw Facts to Meaningful Knowledge

DATA

Raw Facts and Figures

Data is raw, unprocessed facts and figures that have no specific meaning on their own.

EXAMPLES

- 25, 30, 28, 27, 26
- 101, 102, 103, 104
- 01/05/2025, 02/05/2025
- 500, 1500, 2000, 2500

KEY POINTS

- Raw and unorganized
- Without context or meaning
- Large in volume
- Input to processing
- Stored in databases

ANALOGY

Data is like unbaked ingredients (raw rice).

PROCESSING
(Organize, Sort, (Calculate, Analyze)

INFORMATION

Processed and Meaningful

Information is data that has been processed, organized and given meaning to be useful.

EXAMPLES

- Average temperature in May is 27°C
- Total number of students: 4
- Report generated on 02/05/2025
- Total sales amount: ₹5750

KEY POINTS

- Processed and organized
- Has context and meaning
- Useful for decision making
- Output of processing
- Helps in knowledge generation

ANALOGY

Information is like cooked and ready to eat meal.

KEY DIFFERENCE AT A GLANCE

ASPECT	DATA	INFORMATION
Nature	Raw, unprocessed	Processed, meaningful
Meaning	No meaning by itself	Has meaning and context
Use	Input for processing	Used for decision making
Example	25, 30, 28	Average temperature is 27°C
Result	Does not reduce uncertainty	Reduces uncertainty and adds value

SUMMARY

Data is the raw material, while **Information** is the final product. Good information leads to better decisions.