Introduction to DBMS

1.Introduction to SQL

**1. What is SQL, and why is it essential in database management?**

ANS :-

**SQL (Structured Query Language)** is a standard programming language used for managing and manipulating relational databases.  
It is essential because:

* It allows users to create and modify database structures.
* It helps insert, update, delete, and retrieve data efficiently.
* It supports data access control and transaction management.
* SQL is widely used and
* supported by most modern RDBMS like MySQL, PostgreSQL, Oracle, etc.

**2. Explain the difference between DBMS and RDBMS?**

ANS :-

| **Feature** | **DBMS** | **RDBMS** |
| --- | --- | --- |
| Full Form | Database Management System | Relational Database Management System |
| Data Storage | Stores data as files | Stores data in tables (rows and columns) |
| Relationships | No relationships between data | Maintains relationships using foreign keys |
| Data Integrity | Less concern for data integrity | Ensures data integrity and constraints |
| Example | MS Access, file system | MySQL, Oracle, PostgreSQL |

**3. Describe the role of SQL in managing relational databases**

ANS:-

SQL plays a central role in managing relational databases by:

* **Defining schema** using CREATE, ALTER, and DROP.
* **Manipulating data** with INSERT, UPDATE, DELETE, and SELECT.
* **Querying data** with complex conditions, joins, and aggregations.
* **Managing access** through GRANT and REVOKE.
* **Controlling transactions** using COMMIT, ROLLBACK, and SAVEPOINT.

**4. What are the key features of SQL?**

ANS:-

* **Data Definition Language (DDL):** Commands like CREATE, ALTER, DROP.
* **Data Manipulation Language (DML):** Commands like INSERT, UPDATE, DELETE.
* **Data Query Language (DQL):** SELECT for querying data.
* **Data Control Language (DCL):** GRANT, REVOKE for permissions.
* **Transaction Control:** COMMIT, ROLLBACK, SAVEPOINT.
* **Support for joins**, subqueries, indexing, and constraints.

2. SQL Syntax

**1. What are the basic components of SQL syntax?**

ANS :-

The basic components of SQL syntax include:

* **Keywords**: Reserved words like SELECT, FROM, WHERE, INSERT, etc.
* **Identifiers**: Names of databases, tables, columns, or aliases.
* **Operators**: Symbols or words used for comparisons and operations (e.g., =, >, <, LIKE, AND, OR).
* **Literals**: Fixed values such as strings ('text'), numbers (100), or dates ('2025-05-02').
* **Clauses**: Parts of a statement that perform specific functions, like WHERE, ORDER BY, GROUP BY.
* **Functions**: Built-in methods to perform operations (e.g., COUNT(), SUM(), NOW()).
* **Expressions**: Combinations of values, operators, and functions that return a result.

**2. Write the general structure of an SQL SELECT statement.**

ANS:-

sql

CopyEdit

SELECT column1, column2, ...

FROM table\_name

WHERE condition

GROUP BY column

HAVING condition

ORDER BY column ASC|DESC

LIMIT number;

**3. Explain the role of clauses in SQL statements.**

ANS :-

Clauses define the structure and logic of SQL statements. Each clause has a specific role:

* **SELECT**: Specifies the columns to retrieve.
* **FROM**: Indicates the table(s) to query.
* **WHERE**: Filters rows based on conditions.
* **GROUP BY**: Groups rows sharing a common value for aggregate functions.
* **HAVING**: Filters groups created by GROUP BY.
* **ORDER BY**: Sorts the result set.
* **LIMIT**: Restricts the number of returned rows.

3. SQL Constraints

**1. What are constraints in SQL? List and explain the different types of constraints.**

ANS :-

**Constraints** in SQL are rules applied to table columns to enforce data integrity and accuracy.

**Types of constraints:**

* **NOT NULL**: Ensures a column cannot have a NULL value.
* **UNIQUE**: Ensures all values in a column are different.
* **PRIMARY KEY**: Uniquely identifies each row in a table. It combines NOT NULL and UNIQUE.
* **FOREIGN KEY**: Links one table’s column to another table’s PRIMARY KEY to maintain referential integrity.
* **CHECK**: Ensures that all values in a column meet a specified condition.
* **DEFAULT**: Sets a default value for a column when no value is specified.
* **INDEX (implicit)**: While not a constraint, indexes are often created on PRIMARY or UNIQUE keys to improve performance.

**2. How do PRIMARY KEY and FOREIGN KEY constraints differ?**

ANS:-

| **Feature** | **PRIMARY KEY** | **FOREIGN KEY** |
| --- | --- | --- |
| Purpose | Uniquely identifies each row | Establishes a link between two tables |
| Uniqueness | Must be unique | Can have duplicate values |
| NULL values | Cannot contain NULL | Can contain NULL |
| Table location | Defined in the table it belongs to | Refers to another table’s PRIMARY KEY |
| Integrity | Enforces entity integrity | Enforces referential integrity |

**3. What is the role of NOT NULL and UNIQUE constraints?**

ANS :-

* **NOT NULL**:
  + Ensures that a column cannot store NULL values.
  + Guarantees that each row has a value in that column.
  + Used when a field is required (e.g., email, username).
* **UNIQUE**:
  + Ensures that all values in a column are different.
  + Allows NULL values (unlike PRIMARY KEY).
  + Used to avoid duplicate entries (e.g., employee ID, license plate number).

4.MainSQLCommandsandSub-commands(DDL)

**1. Define the SQL Data Definition Language (DDL).**

ANS :-

**Data Definition Language (DDL)** is a subset of SQL used to define and manage the structure of database objects such as tables, views, schemas, and indexes.

**Common DDL commands include:**

* CREATE: To create a new database or table.
* ALTER: To modify an existing database object.
* DROP: To delete an object from the database.
* TRUNCATE: To remove all records from a table, quickly and irreversibly.

**2. Explain the CREATE command and its syntax.**

ANS:-

The CREATE command is used to create new database objects like tables, views, or databases.

**Basic syntax for creating a table:**

sql

CopyEdit

CREATE TABLE table\_name (

column1 datatype [constraint],

column2 datatype [constraint],

...

);

**Example:**

sql

CopyEdit

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Age INT CHECK (Age >= 0),

Email VARCHAR(100) UNIQUE

);

**3. What is the purpose of specifying data types and constraints during table creation?**

ANS :-

Specifying **data types** and **constraints** ensures:

* **Data integrity**: Validates and enforces correct types of data (e.g., INT, VARCHAR).
* **Consistency**: Prevents invalid or duplicate values using constraints like NOT NULL, UNIQUE, and CHECK.
* **Efficiency**: Helps the database engine optimize storage and access.
* **Accuracy**: Reduces chances of data entry errors and maintains reliable relationships between tables.

5. ALTER Command

**1. What is the use of the ALTER command in SQL?**

ANS :-

The ALTER command is used to **change the structure of an existing table**.  
It allows you to:

* **Add** new columns
* **Modify** the data type or constraints of existing columns
* **Drop** (remove) columns
* **Rename** columns or the table (in some databases)
* **Add or drop** constraints (like PRIMARY KEY, FOREIGN KEY)

**2. How can you add, modify, and drop columns from a table using ALTER?**

ANS :-

**a) Add a column:**

sql

CopyEdit

ALTER TABLE Employees

ADD Email VARCHAR(100);

**b) Modify a column:**

sql

CopyEdit

ALTER TABLE Employees

MODIFY Email VARCHAR(150);

**c) Drop a column:**

sql

CopyEdit

ALTER TABLE Employees

DROP COLUMN Email;

6. DROP Command

**1. What is the function of the DROP command in SQL?**

ANS :-

The DROP command is used to **permanently delete** a database object such as a:

* Table
* View
* Index
* Database

**Syntax to drop a table:**

sql

CopyEdit

DROP TABLE table\_name;

Once executed, the object and all its data are **permanently removed**.

**2. What are the implications of dropping a table from a database?**

ANS :-

Dropping a table has the following consequences:

* **Permanent Data Loss**: All data stored in the table is permanently deleted.
* **Loss of Structure**: The table’s schema (columns, data types, constraints) is erased.
* **Dependent Objects Affected**: Views, triggers, stored procedures, or foreign key relationships referencing the table may break or become invalid.
* **No Undo**: Most SQL systems do not support rollback of DROP unless inside a transaction block (and only if supported).

7. Data Manipulation Language (DML)

**1. Define the INSERT, UPDATE, and DELETE commands in SQL.**

ANS :-

**INSERT**

Used to **add new records (rows)** into a table.

**Syntax:**

sql

CopyEdit

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

**UPDATE**

Used to **modify existing records** in a table.

**Syntax:**

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

**DELETE**

Used to **remove records** from a table.

**Syntax:**

sql

CopyEdit

DELETE FROM table\_name

WHERE condition;

**2. What is the importance of the WHERE clause in UPDATE and DELETE operations?**

ANS :-

The WHERE clause is **crucial** because:

* It **specifies which rows** should be updated or deleted.
* **Prevents accidental updates or deletions** of all rows in the table.
* Without a WHERE clause, the command will affect **every row**.

**Example without WHERE:**

DELETE FROM Students;

This will delete **all records** in the Students table.

**Safe example with WHERE:**

DELETE FROM Students

WHERE StudentID = 101;

Deletes **only the specified student**.

8. Data Query Language (DQL)

**1. What is the SELECT statement, and how is it used to query data?**

ANS :-

The SELECT statement is used in SQL to **retrieve data** from one or more tables in a database.

**Basic syntax:**

SELECT column1, column2, ...

FROM table\_name;

SELECT Name, Age

FROM Students;

SELECT \* FROM Students;

The SELECT statement can include additional clauses like WHERE, ORDER BY, GROUP BY, LIMIT, etc., to refine the query results.

**2. Explain the use of the ORDER BY and WHERE clauses in SQL queries.**

ANS :-

**WHERE Clause**

* Filters rows based on a specified condition.
* Only rows **matching the condition** are returned.
* Used with SELECT, UPDATE, and DELETE.

**Example:**

SELECT \* FROM Students

WHERE Age > 18;

Retrieves only students older than 18.

**ORDER BY Clause**

* Sorts the result set by one or more columns.
* Default is **ascending (ASC)** order; can also use **descending (DESC)**.

**Example:**

SELECT \* FROM Students

ORDER BY Age DESC;

Retrieves all students sorted by age in descending order.

**Combined example:**

SELECT Name, Age

FROM Students

WHERE Age >= 18

ORDER BY Name ASC;

Gets students aged 18 or older and sorts them alphabetically by name.

9. Data Control Language (DCL)

**1. What is the purpose of GRANT and REVOKE in SQL?**

ANS :-

* **GRANT** is used to **give permissions** to users or roles to perform specific actions on database objects (e.g., tables, views, procedures).
* **REVOKE** is used to **take back previously granted permissions**.

These commands help in managing **database security** and **user access control**.

**2. How do you manage privileges using these commands?**

ANS :-

**Using GRANT:**

You assign specific privileges (like SELECT, INSERT, UPDATE, etc.) to a user.

**Syntax:**

GRANT privilege\_type ON object\_name TO user\_name;

**Example:**

GRANT SELECT, INSERT ON Students TO user1;

**Using REVOKE:**

You remove previously granted permissions from a user.

**Syntax:**

REVOKE privilege\_type ON object\_name FROM user\_name;

**Example:**

REVOKE INSERT ON Students FROM user1;

**Common Privileges:**

* SELECT – Read data
* INSERT – Add data
* UPDATE – Modify data
* DELETE – Remove data
* ALL – Grants all available privileges

10.Transaction Control Language(TCL)

**1. What is the purpose of the COMMIT and ROLLBACK commands in SQL?**

ANS :-

* **COMMIT**:
  + Saves all changes made during the current transaction **permanently** in the database.
  + Once committed, changes **cannot be undone**.

**Example:**

INSERT INTO Orders (OrderID, Product)

VALUES (101, 'Laptop');

COMMIT;

* **ROLLBACK**:
  + **Undo** all changes made during the current transaction.
  + Useful for cancelling changes due to errors or failed conditions.

**Example:**

DELETE FROM Customers

WHERE CustomerID = 10;

ROLLBACK;

These commands are critical in **maintaining data integrity and consistency**.

**2. Explain how transactions are managed in SQL databases.**

ANS :-

A **transaction** is a sequence of SQL operations executed as a **single logical unit of work**. It must follow the **ACID** properties:

* **A**tomicity: All operations in a transaction succeed, or none do.
* **C**onsistency: The database remains valid before and after the transaction.
* **I**solation: Concurrent transactions do not interfere with each other.
* **D**urability: Once committed, changes are permanent even after a system failure.

**Transaction control commands:**

* BEGIN or START TRANSACTION: Begins a new transaction.
* COMMIT: Saves the transaction.
* ROLLBACK: Cancels the transaction.
* SAVEPOINT: Sets a point within a transaction to rollback to.
* ROLLBACK TO SAVEPOINT: Undoes part of the transaction up to a defined point.

11. SQL Joins

**1. Explain the concept of JOIN in SQL.**

ANS :-

**JOIN** is used in SQL to **combine rows from two or more tables** based on a related column, usually a **foreign key**.

**Types of JOINs and their differences:**

| **Type of JOIN** | **Description** |
| --- | --- |
| **INNER JOIN** | Returns only matching rows from both tables. |
| **LEFT JOIN** | Returns all rows from the **left** table and matching rows from the right. |
| **RIGHT JOIN** | Returns all rows from the **right** table and matching rows from the left. |
| **FULL OUTER JOIN** | Returns all rows from both tables, with NULLs where no match is found |

**2. How are JOINs used to combine data from multiple tables?**

ANS :-

JOINs allow you to **query data across related tables**, such as using foreign keys to connect customers with orders.

**INNER JOIN example:**

SELECT Customers.Name, Orders.Product

FROM Customers

INNER JOIN Orders

ON Customers.CustomerID = Orders.CustomerID;

12. SQL Group By

**1. What is the GROUP BY clause in SQL? How is it used with aggregate functions?**

ANS :-

The **GROUP BY** clause is used to **group rows that have the same values** in specified columns into summary rows.  
It is commonly used with **aggregate functions** like:

* COUNT() – counts rows
* SUM() – adds values
* AVG() – calculates average
* MAX() – finds maximum
* MIN() – finds minimum

**Syntax:**

SELECT column\_name, AGGREGATE\_FUNCTION(column\_name)

FROM table\_name

GROUP BY column\_name;

**Example:**

SELECT Department, COUNT(\*) AS TotalEmployees

FROM Employees

GROUP BY Department;

**2. Explain the difference between GROUP BY and ORDER BY.**

ANS :-

| **Feature** | **GROUP BY** | **ORDER BY** |
| --- | --- | --- |
| Purpose | Groups rows for aggregation | Sorts the result set |
| Usage | Used with aggregate functions | Used to arrange rows by column values |
| Returns | One row per group | All rows (unless combined with limit or filter) |
| Order of data | Not guaranteed (unless ORDER BY used) | Guaranteed sorted output |

13. SQL Stored Procedure

**1. What is a stored procedure in SQL, and how does it differ from a standard SQL query?**

ANS :-

A **stored procedure** is a **precompiled set of one or more SQL statements** that are stored in the database and can be executed repeatedly.

**Key differences from a standard SQL query:**

| **Feature** | **Stored Procedure** | **Standard SQL Query** |
| --- | --- | --- |
| Reusability | Can be called multiple times | Usually written and executed once |
| Logic & Control Flow | Supports conditions, loops, variables | Usually does not include logic |
| Stored in DB | Yes | No |
| Execution | Call using EXEC or CALL | Executed directly |

**Example:**

CREATE PROCEDURE GetEmployeeInfo (@DeptID INT)

AS

BEGIN

SELECT \* FROM Employees WHERE DepartmentID = @DeptID;

END;

**2. Explain the advantages of using stored procedures.**

ANS :-

Here are some key benefits:

* **Improved Performance**: Stored procedures are **precompiled**, so they execute faster than ad-hoc queries.
* **Reusability**: Write once, reuse many times across applications or users.
* **Security**: Users can be granted permission to execute a procedure **without giving direct access** to the underlying tables.
* **Maintainability**: Business logic is centralized in the database, making updates easier.
* **Reduced Network Traffic**: Multiple SQL statements can be executed with a **single call** to the server.

14. SQL View

**1. What is a view in SQL, and how is it different from a table?**

ANS :-

A **view** in SQL is a **virtual table** based on the result of a SELECT query. It does **not store data physically**, but rather presents data from one or more real tables.

**Key differences between a view and a table:**

| **Feature** | **View** | **Table** |
| --- | --- | --- |
| Storage | No physical storage (virtual) | Stores data physically |
| Modifiability | Often read-only (can be updatable) | Fully updatable |
| Basis | Defined by a SELECT query | Defined by schema and holds data |
| Usage | Used for simplified or secured access | Used to store and manage actual data |

**2. Explain the advantages of using views in SQL databases.**

ANS :-

Here are the main benefits:

* **Data Abstraction**: Simplifies complex joins and queries for the end user.
* **Security**: Restricts access to sensitive columns or rows by exposing only what’s needed.
* **Reusability**: Write a complex query once and reuse it like a table.
* **Maintainability**: Changes to query logic can be done in the view without altering application code.
* **Consistency**: Ensures uniform presentation of data to all users.

15. SQL Triggers

**1. What is a trigger in SQL? Describe its types and when they are used.**

ANS :-

A **trigger** is a special kind of stored procedure in SQL that **automatically executes** when a specified **event occurs** on a table or view (like INSERT, UPDATE, or DELETE).

Triggers are used to:

* Enforce business rules
* Maintain data integrity
* Log or audit changes
* Automatically update related tables

**Types of triggers:**

| **Type** | **Description** |
| --- | --- |
| **BEFORE Trigger** | Executes **before** the event occurs (supported in some systems like MySQL). |
| **AFTER Trigger** | Executes **after** the event has occurred. |
| **INSTEAD OF Trigger** | Replaces the event action (mostly used with views). |

**2. Explain the difference between INSERT, UPDATE, and DELETE triggers.**

ANS :-

| **Trigger Type** | **When it Executes** | **Common Uses** |
| --- | --- | --- |
| **INSERT Trigger** | When a new row is added | Validate or log new entries |
| **UPDATE Trigger** | When an existing row is modified | Track changes, enforce update rules |
| **DELETE Trigger** | When a row is deleted | Archive data, prevent deletion in certain conditions |

16. Introduction to PL/SQL

**1. What is PL/SQL, and how does it extend SQL's capabilities?**

ANS :-

**PL/SQL** (Procedural Language/Structured Query Language) is Oracle's **procedural extension** of SQL.  
It allows you to **write code with logical control structures**, such as loops, conditions, and variables — which **standard SQL cannot do**.

While SQL is a **declarative language** focused on data retrieval and manipulation, PL/SQL is a **procedural language** that lets you:

* Declare variables and constants
* Use IF statements, FOR/WHILE loops
* Create functions, procedures, triggers, and packages
* Handle errors with exception blocks

**2. List and explain the benefits of using PL/SQL.**

ANS :-

Here are some key benefits of PL/SQL:

| **Benefit** | **Description** |
| --- | --- |
| **Procedural Logic** | Allows complex business logic using conditions, loops, and control flow. |
| **Better Performance** | Reduces network traffic by sending one block of code instead of many queries. |
| **Reusability** | Supports modular code through stored procedures, functions, and packages. |
| **Security** | Sensitive operations can be done inside procedures without exposing SQL. |
| **Exception Handling** | Provides built-in error handling for runtime issues. |
| **Maintainability** | Centralized business logic makes code easier to manage and update. |

17. PL/SQL Control Structures

**1. What are control structures in PL/SQL?**

ANS :-

Control structures in PL/SQL are programming constructs that allow you to **control the flow of execution** in your code using logic, decisions, and repetition.

**Two key types:**

**IF-THEN structure**

Used to **execute code based on a condition**.

**Syntax:**

IF condition THEN

-- statements

END IF;

**LOOP structure**

Used to **repeat a set of statements**.

**Types of loops:**

* **Basic LOOP**
* **WHILE LOOP**
* **FOR LOOP**

**Example (FOR LOOP):**

FOR i IN 1..5 LOOP

DBMS\_OUTPUT.PUT\_LINE('Iteration: ' || i);

END LOOP;

**2. How do control structures in PL/SQL help in writing complex queries?**

ANS :-

Control structures make PL/SQL much more powerful than plain SQL by allowing you to:

* Perform **conditional logic** (e.g., IF-ELSE, CASE)
* **Iterate** over rows or ranges using loops
* **Handle exceptions** and apply fallback logic
* **Make decisions dynamically** based on data values
* Create **modular**, reusable, and **maintainable** logic

18. SQL Cursors

**1. What is a cursor in PL/SQL?**

ANS :-

A **cursor** is a pointer or handle that allows you to **process query results row by row** in PL/SQL.  
PL/SQL uses **cursors** to retrieve, manipulate, and navigate through query results.

There are **two types** of cursors:

| **Cursor Type** | **Description** |
| --- | --- |
| **Implicit Cursor** | Automatically created by PL/SQL for **single-row** SELECT, INSERT, UPDATE, or DELETE operations. |
| **Explicit Cursor** | Manually declared by the programmer to **fetch multiple rows** one at a time. Used for complex row-by-row processing. |

**2. When would you use an explicit cursor over an implicit one?**

ANS :-

You should use an **explicit cursor** when:

* You need to **process multiple rows** one at a time.
* You want **fine-grained control** over how rows are fetched and processed.
* You need to **loop through a result set** with logic applied to each row.
* The query is **complex**, and you want to **manually open, fetch, and close** the result.

19. Rollback and Commit Savepoint

**1. Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK and COMMIT interact with savepoints?**

ANS :-

A **SAVEPOINT** is a user-defined marker within a transaction that allows you to **partially undo** actions without affecting the entire transaction.

* SAVEPOINT lets you set a point in the transaction you can return to later using ROLLBACK TO.
* ROLLBACK TO savepoint\_name undoes only the changes made **after** the savepoint.
* COMMIT makes **all changes in the transaction permanent**, including those before and after savepoints.
* After a COMMIT, all savepoints in that transaction are lost.

**2. When is it useful to use savepoints in a database transaction?**

ANS :-

You use SAVEPOINT when you want **more control over large or complex transactions**.

**Common use cases:**

* When a transaction includes **multiple steps**, and you might want to **roll back only part** of it.
* For **error recovery**, allowing you to undo just the faulty part without restarting the entire transaction.
* When you need to **test logic or data conditions** mid-transaction before deciding whether to proceed or roll back.
* In **interactive sessions** where user choices determine what to commit or undo.