# **SOFTWARE ENGINEERING ASSIGNMENT**

## **Introduction TO DBMS**

**Introduction to SQL**

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

* **SQL (Structured Query Language)** isa **standard programming language** specifically designed for **managing and manipulating relational databases.** It allows users to **store, retrieve, update, and delete data** in a structured way.
* **Why SQL Is Essential in Database Management**
* **SQL** is the **foundation of relational database management**, essential for querying, organizing, and securing data efficiently across nearly all data-driven systems.

1. **Explain the difference between DBMS and RDBMS.**

* **Definition:**

**DBMS:** A database management system in software that allows user to store retrieve, and manage data in data base.

**RDBMS:** A Relational Database Management System is an advanced type of DBMS that stores data in related tables following E. F. Codd’s relational model.

* **Data storage structure:**

**DBMS:** Data is stored as **files** or **non-relational forms** (e.g., hierarchical or network models).

**RDBMS:** Data is stored in **tables (rows and columns)**, and relationships between tables are established using **foreign keys**.

* **Relationships Between Data**

**DBMS:** No relationships between data; data is generally **independent**.

**RDBMS:** Relationships exist between tables using **primary keys and foreign keys.**

* **Data Integrity:**

**DBMS:** Does **not enforce data integrity constraints** (like unique, not null, foreign key, etc.).

**RDBMS:** **Enforces data integrity** and maintains **ACID properties** (Atomicity, Consistency, Isolation, Durability).

* **Normalization**

**DBMS:** Normalization is not supported.

**RDBMS:** Supports data normalization to minimize redundancy.

* **Example Systems**

**DBMS Examples:** Microsoft Access, data BASE, File System, FoxPro

**RDBMS Examples:** MySQL, Oracle, PostgreSQL, Microsoft SQL Server, IBM DB2

* **Multi-user Support**

**DBMS:** Usually supports a single user at a time.

**RDBMS:** Supports multiple concurrent users.

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

* SQL defines the structure and organization of data within the database.
* It allows you to create, alter, and delete database objects like tables, schemas, indexes, and views.

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

* SQL’s key features—such as data definition, manipulation, querying, and integrity control—make it the most important tool for managing relational databases efficiently and securely.

SQL (Structured Query Language) is a standard language used to manage and manipulate relational databases. It provides several key features that make it essential for database management:

* **Data Definition:**  
  SQL allows users to define and modify the structure of a database using commands like CREATE, ALTER, and DROP.
* **Data Manipulation:**  
  It enables inserting, updating, deleting, and retrieving data using commands such as INSERT, UPDATE, DELETE, and SELECT.
* **Data Querying:**  
  SQL can retrieve specific information from large databases using powerful query capabilities like filtering, sorting, and grouping.
* **Data Control:**  
  It provides commands like GRANT and REVOKE to control user access and maintain database security.
* **Transaction Control:**  
  SQL ensures data consistency through transactions using commands such as COMMIT, ROLLBACK, and SAVEPOINT.
* **Data Integrity:**  
  It maintains data accuracy and consistency through constraints like PRIMARY KEY, FOREIGN KEY, NOT NULL, and UNIQUE.
* **Support for Relationships:**  
  SQL can link multiple tables using relationships (joins) to organize related data effectively.
* **Standardization and Portability:**  
  SQL is an ANSI/ISO standard language and works across most database systems like MySQL, Oracle, and SQL Server.

**SQL Syntax**

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

**->** The basic components of SQL syntax are the fundamental elements used to write SQL commands that interact with a database. These include:

1. **Keywords**: Reserved words used to perform specific actions (e.g., SELECT, FROM, WHERE).
2. **Identifiers**: Names given to database objects like tables, columns, or databases.
3. **Clauses**: Parts of a statement that define conditions or specify actions (e.g., WHERE, GROUP BY).
4. **Expressions**: Combinations of values, operators, and functions that return a value.
5. **Predicates**: Conditions used to filter data (e.g., =, <, >).
6. **Statements**: Complete SQL commands like SELECT, INSERT, UPDATE, and DELETE.

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

-> The SELECT statement is used to retrieve data from one or more tables. Its general structure is:

SELECT column1, column2, ...

FROM table\_name

WHERE condition

GROUP BY column\_name

HAVING condition

ORDER BY column\_name;

-> **Explanation of parts:**

* **SELECT:** Specifies the columns to be displayed.
* **FROM:** Indicates the table(s) from which data is retrieved.
* **WHERE:** Filters rows based on a condition.
* **GROUP BY:** Groups rows that have the same values.
* **HAVING:** Filters groups after grouping is applied.
* **ORDER BY:** Sorts the results in ascending or descending order.

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

**->** Clauses are components or building blocks of SQL statements that define specific actions or conditions. Each clause serves a particular purpose in shaping the query.

**Roles of common clauses:**

* **SELECT clause:** Specifies which columns of data to retrieve.
* **FROM clause:** Identifies the table(s) containing the data.
* **WHERE clause:** Filters rows based on given conditions.
* **GROUP BY clause:** Groups rows that have the same values in specified columns.
* **HAVING clause:** Filters grouped data.
* **ORDER BY clause**: Determines the order of the query results.

**In summary:** Clauses make SQL statements structured, readable, and powerful, allowing users to extract and manage data precisely.

**SQL Constraints**

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

**->** Constraints in SQL are rules applied to table columns to ensure the accuracy, validity, and integrity of data in a database.  
Types of constraints:

1. NOT NULL: Ensures that a column cannot have a NULL value.
2. UNIQUE: Ensures all values in a column are distinct.
3. PRIMARY KEY: Uniquely identifies each record in a table (combines NOT NULL and UNIQUE).
4. FOREIGN KEY: Establishes a relationship between two tables by linking columns.
5. CHECK: Ensures that data in a column meets a specific condition.
6. DEFAULT: Assigns a default value when no value is provided.

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

**->** A PRIMARY KEY uniquely identifies each record in a table and cannot contain NULL values.

-> A FOREIGN KEY is a field in one table that refers to the PRIMARY KEY in another table, establishing a relationship between the two tables.

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

-> NOT NULL: Ensures a column always contains a value. It prevents missing data.

-> UNIQUE: Ensures all entries in a column are different, maintaining data uniqueness.

**Main SQL Commands and Sub-commands(DDL)**

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

-> DDL (Data Definition Language) is a set of SQL commands used to define, create, and modify database structures, such as tables, schemas, and indexes.  
Common DDL commands: CREATE, ALTER, DROP, TRUNCATE.

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

**->** The CREATE command is used to create new database objects like tables or views.

-> Syntax:

**CREATE TABLE table\_name (**

**column1 datatype constraint,**

**column2 datatype constraint,**

**...**

**);**

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

-> Datatypes define the type of data (e.g., INT, VARCHAR) that can be stored, while constraints enforce rules on the data (e.g., NOT NULL, UNIQUE).  
They ensure data validity, accuracy, and consistency.

**ALTER Command**

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

-> The ALTER command is used to modify an existing table’s structure, such as adding, changing, or deleting columns and constraints.

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

-> ALTER TABLE table\_name ADD column\_name datatype; -- Add a column

ALTER TABLE table\_name MODIFY column\_name datatype; -- Modify column datatype

ALTER TABLE table\_name DROP COLUMN column\_name; -- Drop a column

**DROP Command**

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

-> The DROP command permanently deletes a database object such as a table, view, or index from the database.

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

-> When a table is dropped, all its data, structure, and relationships are permanently deleted and cannot be recovered unless backed up.

**DataManipulationLanguage(DML)**

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

* **INSERT: Adds new rows into a table.**

Example: INSERT INTO Students VALUES (1, 'John', 20);

* **UPDATE**: Modifies existing data in a table.

UPDATE Students SET Age = 21 WHERE StudentID = 1;

* **DELETE:** Removes rows from a table.

DELETE FROM Students WHERE StudentID = 1;

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

-> The WHERE clause specifies which records should be affected. Without it, all rows in the table will be updated or deleted.

**Data Query Language(DQL)**

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

**->** -> The SELECT statement retrieves data from one or more tables.

-Example:

SELECT Name, Age FROM Students WHERE Age > 18;

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

* **-> WHERE:** Filters records based on condition**s.**
* **ORDER BY**: Sorts the result in ascending (ASC) or descending (DESC) order**.**

**Data Control Language (DCL)**

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

-> **GRANT**: Gives specific privileges to users.

-> **REVOKE**: Removes previously granted privileges.

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

**->** Privileges are managed by assigning or removing access:

GRANT SELECT ON Students TO user1;

REVOKE SELECT ON Students FROM user1;

**Transaction ControlLanguage (TCL)**

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

**-> Purpose of COMMIT and ROLLBACK:**

* **COMMIT:** Saves all changes made in the current transaction permanently to the database. Once committed, changes cannot be undone.
* **ROLLBACK:** Undoes all changes made in the current transaction, restoring the database to its last committed state.

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

-> A transaction is a sequence of SQL operations treated as a single logical unit of work.

**-> SQL ensures ACID properties:**

**Atomicity** – All changes happen completely or not at all.

**Consistency** – Database remains in a valid state before and after the transaction.

**Isolation** – Concurrent transactions don’t interfere with each other.

**Durability** – Once committed, changes are permanent.

**SQL Joins**

1. **Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?**

**->** JOIN is used to combine rows from two or more tables based on a related column.

Types of Joins:

**INNER JOIN:** Returns rows where there is a match in both tables.

**LEFT JOIN (LEFT OUTER JOIN):** Returns all rows from the left table and matched rows from the right table. Unmatched right rows are NULL.

**RIGHT JOIN (RIGHT OUTER JOIN):** Returns all rows from the right table and matched rows from the left table. Unmatched left rows are NULL.

**FULL OUTER JOIN:** Returns all rows from both tables, matching where possible and filling NULL where there’s no match.

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

**->** Joins use common key columns (like customer\_id, dept\_id, etc.) to merge related data from multiple tables into a single result set, allowing queries that span multiple entities.

**SQL Group By**

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

-> The GROUP BY clause groups rows that have the same values in specified columns into summary rows.

-> It is used with aggregate functions (like COUNT(), SUM(), AVG(), MAX(), MIN()) to perform calculations on each group.

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

->GROUP BY: Groups rows based on column values for aggregation.

-> ORDER BY: Sorts the result set in ascending or descending order.

-> GROUP BY changes the structure of the result; ORDER BY only changes the display order.

**SQL Stored Procedure**

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

-> A stored procedure is a precompiled collection of SQL statements stored in the database that can be executed as a single unit.

-> It differs from a regular SQL query because it can include control logic, parameters, and can be reused multiple times.

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

* **Advantages of using procedures:**

-> Improved performance (precompiled execution)

-> Code reusability and modularity

-> Enhanced security (controlled access)

-> Reduced network traffic (less data sent over connections)

-> Easier maintenance (centralized logic)

**SQL View**

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

-> A view is a virtual table created by a query that displays data from one or more tables.

-> Unlike a table, a view doesn’t store data physically; it just stores the SQL query.

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

-> Simplifies complex queries

-> Provides data security (restricts access to specific columns/rows)

-> Offers data abstraction and consistency

-> Can be used to present computed or derived data

**SQL Triggers**

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

-> A trigger is a set of SQL statements that automatically execute in response to specific database events (INSERT, UPDATE, DELETE).

**Types of Triggers:**

* BEFORE Trigger – Executes before the triggering event.
* AFTER Trigger – Executes after the event.
* INSTEAD OF Trigger – Executes instead of the event (mainly for views).

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

**-> INSERT Trigger –** Fires when a new record is inserted.

**-> UPDATE Trigger –**Fires when an existing record is modified.

**-> DELETE Trigger –** Fires when a record is deleted.

Each can be used to enforce business rules, maintain audit logs, or ensure data integrity.

**Introduction to PL/SQL**

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

-> PL/SQL (Procedural Language/SQL) is Oracle’s procedural extension to SQL.

-> It extends SQL by adding programming constructs like variables, loops, conditions, and exception handling — enabling procedural logic in database programs.

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

-> Modularity through procedures, functions, and packages

-> Improved performance via block execution

-> Error handling with exceptions

-> Integration of SQL and procedural logic

->Security and code reuse

**PL/SQL Control Structures**

1. **What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.**

-> Control structures determine the flow of execution in PL/SQL.

-> I**F THEN** -: Executes statements conditionally.

**Example:**

IF salary > 5000 THEN

bonus := 1000;

END IF;

->**LOOP**: Repeats a block of code multiple times.

**Example:**

LOOP

counter := counter + 1;

EXIT WHEN counter > 10;

END LOOP;

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

**->** They allow conditional logic and iterative execution, enabling developers to perform advanced computations, validations, and batch operations directly within the database.

**SQL Cursors**

1. **What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.**

**->** A cursor is a pointer to a result set of a query.

**-> Implicit Cursor:** Automatically created for single-row queries (like SELECT INTO).

**-> Explicit Cursor:** Defined manually for multi-row queries using OPEN, FETCH, and CLOSE statements.

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

**→  
Use an explicit cursor when:**

* The query returns multiple rows.
* You need to process each row individually.
* You need better control over row fetching and processing.

**Rollback and Commit Save point**

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

**->** A SAVEPOINT marks a point within a transaction that you can roll back to without affecting previous work.

-> ROLLBACK TO SAVEPOINT undoes changes after that save point.

-> COMMIT makes all changes (including those before and after save points) permanent.

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

**->** Save points are useful when:

* You want partial rollbacks within a large transaction.
* You need to test intermediate operations before committing.
* You’re performing multiple related changes but want control over specific sections.