……Basic Level……..……

1. What is SQL? Explain its purpose.

SQL (Structured Query Language) is a standardized language used to communicate with and manipulate databases. It is used for querying, updating, and managing data in relational database management systems (RDBMS).

2. What are the different types of SQL commands? (DML, DDL, DCL, TCL, and DQL)

- **DML (Data Manipulation Language):** Commands that manipulate data in tables (e.g., `SELECT`, `INSERT`, `UPDATE`, `DELETE`).
- **DDL (Data Definition Language):** Commands that define the structure of database objects (e.g., `CREATE`, `ALTER`, `DROP`).
- **DCL (Data Control Language):** Commands that control access to data (e.g., `GRANT`, `REVOKE`).
- **TCL (Transaction Control Language):** Commands that manage transactions (e.g., `COMMIT`, `ROLLBACK`, `SAVEPOINT`).
- **DQL (Data Query Language):** Primarily the `SELECT` command, used to query data from the database.

3. What is a Database? What are Tables in SQL?

- **Database:** A collection of organized data that can be easily accessed, managed, and updated.
- **Tables:** Structures within a database that store data in rows and columns. Each table represents a specific entity, such as customers or orders.

4. What is a Primary Key?

A Primary Key is a column (or a set of columns) in a table that uniquely identifies each row. It must contain unique values and cannot contain NULL values.

5. What is a Foreign Key?

A **Foreign Key** is a column (or set of columns) in one table that refers to the Primary Key in another table. It is used to maintain referential integrity between the two tables.

6. What is a Unique Key? How is it different from a Primary Key?

A Unique Key ensures all values in a column are unique across the table, but unlike a Primary Key, it can contain NULL values. A table can have multiple Unique Keys, but only one Primary Key.

7. What is a NULL value in SQL?

A **NULL** value represents missing, unknown, or inapplicable data. It is not the same as zero or an empty string and indicates the absence of a value.

8. What is a Self Join?

A Self Join is a join where a table is joined with itself. It can be used to compare rows within the same table.

9. What are Constraints in SQL? Name some commonly used constraints.

Constraints are rules enforced on data columns in a table to ensure data integrity and accuracy. Common constraints include:

- ❖ PRIMARY KEY
- ❖ FOREIGN KEY
- ❖ UNIQUE
- ❖ NOT NULL
- ❖ CHECK
- ❖ DEFAULT

10. What are Joins in SQL? Describe different types of Joins. (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN)

Joins are used to combine rows from two or more tables based on a related column:

- ❖ INNER JOIN: Returns rows with matching values in both tables.
- ❖ LEFT JOIN (LEFT OUTER JOIN): Returns all rows from the left table and matched rows from the right table; fills with NULLs if no match.
- ❖ RIGHT JOIN (RIGHT OUTER JOIN): Returns all rows from the right table and matched rows from the left table; fills with NULLs if no match.
- ❖ FULL JOIN (FULL OUTER JOIN): Returns rows when there is a match in either table; fills with NULLs for non-matching rows.
- ❖ CROSS JOIN: Returns the Cartesian product of both tables (every combination of rows).

……Intermediate Level…..……

1) What is a Subquery? Describe its types. (Single-row, Multi-row, and Correlated Subqueries)

A **Subquery** is a query nested inside another query (like SELECT, INSERT, UPDATE, or DELETE). It can return data to the main query.

- **Single-row Subquery:** Returns one row of data.
- **Multi-row Subquery:** Returns multiple rows.
- **Correlated Subquery:** Depends on the outer query; it is executed repeatedly for each row processed by the outer query.

2) What is the difference between WHERE and HAVING clause?

- ❖ **WHERE Clause:** Filters rows before grouping and is used with SELECT, UPDATE, DELETE.
- ❖ **HAVING Clause:** Filters groups after aggregation and is used with GROUP BY.

3) Explain GROUP BY and ORDER BY clause. How do they differ?

- **GROUP BY:** Groups rows that have the same values in specified columns and is often used with aggregate functions (SUM, COUNT).
- **ORDER BY:** Sorts the result set based on one or more columns, either ascending or descending.

Difference: GROUP BY organizes rows into groups, while ORDER BY sorts the results.

4) What is an Index? What are the types of indexes?

An **Index** is a database object that improves the speed of data retrieval operations on a table at the cost of additional storage and maintenance overhead.

Types of Indexes:

- Primary Index
- Unique Index
- Composite Index (multiple columns)
- Full-Text Index
- Clustered Index (reorders physical data)
- Non-Clustered Index (logical order, independent of physical order)

5) What are Stored Procedures and Functions in SQL? How do they differ?

- **Stored Procedure:** A reusable set of SQL statements that perform a specific task, can take input parameters, and may return results.
- **Function:** A set of SQL statements that perform calculations or other tasks and return a single value. Functions can be used in SQL queries.

6) What is a Trigger in SQL? When would you use one?

A **Trigger** is a set of SQL statements that automatically execute in response to certain events on a table or view (e.g., INSERT, UPDATE, DELETE). They are used for enforcing business rules, maintaining audit trails, and automating complex tasks.

7) What is Normalization? Describe the normal forms.

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity.

Normal Forms:

- **1NF** (First Normal Form): Eliminate duplicate columns; each column should contain atomic values.
- **2NF** (Second Normal Form): Remove partial dependencies; every non-key column should be fully dependent on the primary key.
- **3NF** (Third Normal Form): Remove transitive dependencies; non-key columns should not depend on other non-key columns.
- **BCNF** (Boyce-Codd Normal Form): A stricter version of 3NF where every determinant is a candidate key.

8) What is Denormalization? When would you use it?

Denormalization is the process of combining normalized tables to reduce the complexity of queries and improve read performance by introducing redundancy. It is used when read performance is prioritized over storage efficiency, such as in reporting and analytical databases.

9) What are ACID properties in a database? Explain each one.

ACID stands for **Atomicity, Consistency, Isolation, and Durability**, which are key properties that ensure reliable transactions in a database:

- **Atomicity:** Ensures that all parts of a transaction are completed; if any part fails, the entire transaction is rolled back.
- **Consistency:** Ensures that a transaction brings the database from one valid state to another, maintaining all rules and constraints.
- **Isolation:** Ensures that transactions are executed independently without interference, appearing as if they are executed sequentially.
- **Durability:** Ensures that once a transaction is committed, its changes are permanent, even in the case of a system failure.

10) What is a View? Explain its use.

A View is a virtual table that is based on the result set of a SQL query. It is used to simplify complex queries, provide a layer of security by restricting access to specific data, and present data in a specific format.

……Advance Level……..

1. What are Transactions in SQL? How do you ensure they are executed properly?

A Transaction in SQL is a sequence of one or more SQL operations that are treated as a single unit of work. A transaction ensures that the operations are completed fully; if any part fails, the entire transaction is rolled back.

To ensure transactions are executed properly:

- ❖ Use ACID properties (Atomicity, Consistency, Isolation, Durability).
- ❖ Use COMMIT to save changes when all operations succeed.
- ❖ Use ROLLBACK to undo changes if any operation fails.

2. Explain the difference between COMMIT and ROLLBACK.

- ❖ **COMMIT:** Saves all changes made during the current transaction and makes them permanent.
- ❖ **ROLLBACK:** Reverts all changes made during the current transaction, undoing any modifications since the last COMMIT.

3. What is the difference between DELETE, TRUNCATE, and DROP commands?

- **DELETE:** Removes specified rows from a table; can be filtered with a WHERE clause. It logs individual row deletions and can be rolled back.
- **TRUNCATE:** Removes all rows from a table without logging individual row deletions; it cannot be rolled back.
- **DROP:** Deletes the entire table or database, including its structure; it is irreversible.

4. What are Cursors in SQL? When should they be used?

A Cursor is a database object used to retrieve and manipulate row-by-row results from a query. They are used when:

- You need to process rows individually.
- You require complex row-by-row processing that cannot be done with standard SQL.

5. Explain the concept of Referential Integrity in SQL.

Referential Integrity ensures that relationships between tables remain consistent. It prevents actions that would leave orphaned records by enforcing rules with Foreign Keys, ensuring that a referenced record in one table exists in the related table.

6. What are Aggregate Functions in SQL? Give examples.

Aggregate Functions perform calculations on a set of values and return a single result. Common examples include:

- SUM(): Calculates the total of a numeric column.
- AVG(): Calculates the average value of a numeric column.
- COUNT(): Counts the number of rows.
- MAX(): Finds the maximum value in a column.
- MIN(): Finds the minimum value in a column.

7. What is the difference between Scalar and Aggregate Functions?

- **Scalar Functions:** Operate on a single value and return a single value (e.g., UCASE(), LENGTH()).
- **Aggregate Functions:** Operate on multiple values and return a single aggregated value (e.g., SUM(), AVG()).

8. What is a Data Warehouse? How is it related to SQL?

A Data Warehouse is a centralized repository that stores large volumes of structured data from multiple sources, optimized for query and analysis. SQL is commonly used in data warehouses for querying, managing, and analyzing stored data.

9. Explain SQL Injection and how it can be prevented.

SQL Injection is a security vulnerability that allows attackers to manipulate SQL queries by injecting malicious SQL code, potentially accessing or modifying the database.

Prevention:

- Use Parameterized Queries or Prepared Statements.
- Employ Input Validation to filter out suspicious input.
- Use Stored Procedures to separate user input from SQL logic.
- Implement Least Privilege access to database users.

10. What are Common Table Expressions (CTE)? How are they different from Subqueries?

A Common Table Expression (CTE) is a temporary result set defined within the execution scope of a single SQL statement using the WITH clause.

Differences from Subqueries:

- CTEs are more readable and reusable within a query.
- They can be referenced multiple times in the same query.
- CTEs can perform recursive operations, which subqueries cannot do directly.

11. What is a Composite Key?

A Composite Key is a key that consists of two or more columns in a table that together uniquely identify a row. It is used when a single column is not sufficient to ensure uniqueness.

12. What is the difference between Cross Join and Natural Join?

- **Cross Join:** Produces the Cartesian product of two tables, returning all possible combinations of rows from both tables.
- **Natural Join:** Combines rows from two tables based on columns with the same name and data type, automatically matching rows with equal values in those columns.

13. What is the use of the UNION and UNION ALL operators?

- **UNION:** Combines the result sets of two or more SELECT statements, removing duplicate rows.
- **UNION ALL:** Combines the result sets of two or more SELECT statements, including all duplicate rows.

14. What are some ways to optimize SQL queries?

- **Use Indexes:** To speed up data retrieval.
  Avoid Select : Specify only needed columns to reduce data load.
- **Use Joins Efficiently:** Use appropriate joins (e.g., INNER JOIN) and avoid unnecessary joins.
- **Limit Result Sets:** Use LIMIT or TOP to fetch only required rows.
- **Optimize WHERE Clauses:** Use indexed columns in WHERE clauses and avoid functions on columns.
- **Use Proper Data Types:** Choose appropriate data types for columns to reduce storage and improve performance.
- **Analyze Execution Plans:** To identify and optimize slow parts of the query.

15. What is Database Indexing? How does it improve query performance?

Database Indexing is a technique used to improve the speed of data retrieval operations on a database table by creating a data structure (index) that allows quick lookup of rows.

Performance Improvement:

- Indexes provide a fast path to find data without scanning the entire table.
- They reduce the number of rows the database engine needs to examine.
- However, indexing can slow down write operations (INSERT, UPDATE, DELETE) due to index maintenance overhead.