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

* SQL or simply “sequel,” is a programming language that communicates through relational databases. It is the easiest way to store, update, remove, search, or retrieve information on a database.
* SQL allows developers to easily manipulate data at many different sizes as their needs change, without needing to rewrite their entire application.

1. Explain the difference between DBMS and RDBMS.

* A DBMS (Database Management System) is a software application that manages a database, providing features for creating, updating, and accessing data, while an RDBMS (Relational Database Management System) is a specific type of DBMS that organizes data into tables with relationships. RDBMS uses the relational model, where data is structured into tables with rows and columns, and relationships between tables are enforced through keys like primary and foreign keys.

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

* Relational databases, enabling users to create, query, update, and delete data within these databases. It provides a standard way for applications and users to interact with relational database management systems (RDBMS) like MySQL, PostgreSQL, Oracle, and SQL Server.
* Key Roles of SQL in Managing Relational Databases:
* Data Manipulation: SQL allows users to retrieve, insert, modify, and delete data within the database tables. For example, using SELECT statements to query data, INSERT to add new records, UPDATE to modify existing ones, and DELETE to remove data.
* Data Definition: SQL is used to define the structure of the database, including creating tables, defining relationships between them (like primary and foreign keys), and setting constraints on data.
* Access Control and Security: SQL provides mechanisms to control who can access the database and what they can do with the data. This includes setting permissions and creating user roles with specific privileges, according to Quora.
* Data Integrity: SQL helps ensure the accuracy and consistency of the data within the database by enforcing constraints and relationships, says IBM.
* Application Integration: SQL allows applications to easily connect to and interact with relational databases, providing a way to store, retrieve, and manipulate data needed for various functionalities, states Amazon Web Services (AWS).

1. What are the key features of SQL?

* Key features of SQL: Data Definition Language (DDL): SQL provides many commands, one of which is Data Definition Language (DDL). ...
* Data Manipulation Language (DML) ...
* Relational Foundation. ...
* High-performance. ...
* Scalability. ...
* Security and authentication. ...
* Vendor Independence. ...
* Portability across different computer systems

1. What are the basic components of SQL syntax?

* The primary components include tables, queries, clauses, the SELECT statement, the INSERT command, data types, and expressions.
* Additionally, SQL is divided into three main components: Data Definition Language (DDL), Data Control Language (DCL), and Data Manipulation Language (DML).

1. Write the general structure of an SQL SELECT statement.

* The general structure of an SQL SELECT statement follows this format:
* SELECT column1, column2, ... FROM table\_name [WHERE condition] [ORDER BY column\_name [ASC|DESC]].
* The SELECT keyword specifies which columns to retrieve, FROM indicates the table to query, WHERE filters rows based on a condition, and ORDER BY sorts the results.

1. Explain the role of clauses in SQL statements

* SQL clauses are built-in functions that define specific conditions within an SQL statement to retrieve, update, or manipulate data from a database. These clauses work alongside SELECT, UPDATE, DELETE, and INSERT queries to refine results and ensure efficient data handling.

Key Functions of SQL Clauses:

* Filter and retrieve specific records from a database.
* Group data based on certain attributes.
* Sort query results in ascending or descending order.
* Limit the number of records displayed in a query result.

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

* SQL constraints are rules applied to columns or tables in a relational database to limit the type of data that can be inserted, updated, or deleted. These rules ensure the data is valid, consistent, and adheres to the business logic or database requirements. Constraints can be enforced during table creation or later using the ALTER TABLE statement. They play a vital role in maintaining the quality and integrity of your database.
* Types of SQL Constraints:
* SQL provides several types of constraints to manage different aspects of data integrity.
* 1. NOT NULL Constraint:-
* The NOT NULL constraint ensures that a column cannot contain NULL values. This is particularly important for columns where a value is essential for identifying records or performing calculations. If a column is defined as NOT NULL, every row must include a value for that column.

Example:

CREATE TABLE Student

(

ID int(6) NOT NULL,

NAME varchar(10) NOT NULL,

ADDRESS varchar(20)

);

* 2. UNIQUE Constraint:-
* The UNIQUE constraint ensures that all values in a column are distinct across all rows in a table. Unlike the PRIMARY KEY, which requires uniqueness and does not allow NULLs, the UNIQUE constraint allows NULL values but still enforces uniqueness for non-NULL entries.

Example:

CREATE TABLE Student

(

ID int(6) NOT NULL UNIQUE,

NAME varchar(10),

ADDRESS varchar(20)

);

* 3. PRIMARY KEY Constraint :-
* A PRIMARY KEY constraint is a combination of the NOT NULL and UNIQUE constraints. It uniquely identifies each row in a table. A table can only have one PRIMARY KEY, and it cannot accept NULL values. This is typically used for the column that will serve as the identifier of records.

Example:

CREATE TABLE Student

(

ID int(6) NOT NULL UNIQUE,

NAME varchar(10),

ADDRESS varchar(20),

PRIMARY KEY(ID)

);

4. FOREIGN KEY Constraint:-

* A FOREIGN KEY constraint links a column in one table to the primary key in another table. This relationship helps maintain referential integrity by ensuring that the value in the foreign key column matches a valid record in the referenced table.

Example:

CREATE TABLE Orders

(

O\_ID int NOT NULL,

ORDER\_NO int NOT NULL,

C\_ID int,

PRIMARY KEY (O\_ID),

FOREIGN KEY (C\_ID) REFERENCES Customers(C\_ID)

)

* 5. CHECK Constraint:
* The CHECK constraint allows us to specify a condition that data must satisfy before it is inserted into the table. This can be used to enforce rules, such as ensuring that a column’s value meets certain criteria (e.g., age must be greater than 18)
* Example:

CREATE TABLE Student

(

ID int(6) NOT NULL,

NAME varchar(10) NOT NULL,

AGE int NOT NULL CHECK (AGE >= 18)

);

* 6. DEFAULT Constraint
* The DEFAULT constraint provides a default value for a column when no value is specified during insertion. This is useful for ensuring that certain columns always have a meaningful value, even if the user does not provide one

Example:

CREATE TABLE Student

(

ID int(6) NOT NULL,

NAME varchar(10) NOT NULL,

AGE int DEFAULT 18

);

1. How do PRIMARY KEY and FOREIGN KEY constraints differ?

* Primary and foreign keys are both crucial for database integrity, but they serve different purposes. A primary key uniquely identifies each row within a table, ensuring data uniqueness and serving as the foundation for relationships with other tables.
* A foreign key, on the other hand, establishes a link between two tables by referencing the primary key of another table, enforcing referential integrity and allowing for data relationships.

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

* NOT NULL constraints prevent null values from being entered into a column. Unique constraints ensure that the values in a set of columns are unique and not null for all rows in the table. The columns specified in a unique constraint must be defined as NOT NULL.

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

* In the context of SQL, data definition or data description language is a syntax for creating and modifying database objects such as tables, indices, and users. DDL statements are similar to a computer programming language for defining data structures, especially database schemas.

1. Explain the CREATE command and its syntax.

* The CREATE command in SQL is a Data Definition Language (DDL) command used to create new database objects, such as tables, databases, views, and indexes. It defines the structure of these objects, including their names, columns, data types, and constraints.
* Syntax:
* The general syntax for creating a database object using the CREATE command varies slightly depending on the object type, but it often follows this pattern:

Code

CREATE <object\_type> <object\_name> (

<column1\_name> <data\_type> [constraints],

<column2\_name> <data\_type> [constraints],

    ...

);

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

* Specifying data types and constraints during table creation in a database ensures data integrity, accuracy, and consistency. Data types define the kind of data a column can hold (e.g., text, numbers, dates), while constraints enforce rules on the data within a table, like preventing null values or ensuring uniqueness.

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

* The ALTER TABLE can be used in order to add, drop, delete, or modify the columns in an already existing table. This statement can also be used to add or drop multiple constraints on an already existing table

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

* SQL ALTER TABLE STATEMENT
* table\_name refers to the name of the table you want to modify.
* ADD is used to add a new column.
* DROP is used to remove an existing column.
* MODIFY is used to change the datatype or definition of an existing column.

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

* The DROP command in SQL is used to permanently delete a database object, such as a table, view, or index, from a database. When a table is dropped, both its structure (schema) and data are removed, and this action is irreversible.
* Dropping a table means losing all data stored in the table, along with any associated objects like indexes, triggers, and constraints.

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

* Dropping a table in a database has significant implications, including permanent data loss, removal of the table structure, and potential impact on dependent objects. It is a destructive operation that should be carefully considered before execution.

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

* SQL supports three primary data manipulation operations:
* INSERT: This operation allows you to add new records or rows to a table.
* UPDATE: The UPDATE operation enables you to modify existing records in a table.
* DELETE: The DELETE operation allows you to remove records from a table.

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

* Whether you are retrieving data, updating records, or deleting entries from a database, the WHERE clause plays an important role in defining which rows will be affected by the query. Without it, SQL queries would return all rows in a table, making it difficult to target specific data.

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

* The SELECT statement in SQL is used to query data from a database by retrieving specific columns from one or more tables. It is a fundamental SQL statement for data retrieval, allowing you to specify exactly which data you want to see.

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

* In SQL, the WHERE clause is used to filter rows based on specific conditions, while the ORDER BY clause sorts the result set based on one or more columns. The WHERE clause comes before the ORDER BY clause in the query execution order.

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

* GRANT & REVOKE are the popular members of the SQL family. These are the types of DCL commands that are used to assign permission to the users to perform a different task. The GRANT command is used for permitting the users whereas the REVOKE command is used for removing the authorization.

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

* The COMMIT statement lets a user save any changes or alterations on the current transaction. These changes then remain permanent.
* The ROLLBACK statement lets a user undo all the alterations and changes that occurred on the current transaction after the last COMMIT.

1. Explain how transactions are managed in SQL databases.

* In SQL, transaction control commands manage the execution of SQL operations, ensuring the integrity and reliability of database transactions. These commands help manage the start, commit, and rollback of changes made to the database.

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

* In SQL, a JOIN clause combines rows from two or more tables based on a related column between them.
* There are four main types of joins: INNER JOIN, LEFT JOIN (or LEFT OUTER JOIN), RIGHT JOIN (or RIGHT OUTER JOIN), and FULL OUTER JOIN.
* INNER JOIN returns only the rows where there is a match in both tables, while OUTER JOINs (LEFT, RIGHT, and FULL) return all rows from one or both tables, including unmatched rows, which are filled with NULL values.

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

* A JOIN is a means for combining columns from multiple tables by using values common to each. The JOIN keyword combined with ON is used to combine fields from separate tables. A JOIN clause on its own will result in a cross product, where each row in the first table is paired with each row in the second table.

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

* The GROUP BY clause splits the result-set into groups of values and the aggregate function can be used to return a single value for each group.
* The most commonly used SQL aggregate functions are: MIN() - returns the smallest value within the selected column. MAX() - returns the largest value within the selected column.

1. Explain the difference between GROUP BY and ORDER BY.

* In SQL, GROUP BY groups rows with identical values in one or more columns, while ORDER BY sorts the result set of a query in ascending or descending order based on one or more columns.
* GROUP BY is often used with aggregate functions (like COUNT(), SUM(), AVG()) to perform calculations on each group. ORDER BY simply arranges the results without changing the underlying data structure.

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

* A stored procedure in SQL is a pre-compiled collection of SQL statements stored within a database, while a standard SQL query is a single, ad-hoc SQL statement executed on demand.
* Stored procedures offer benefits like reusability, performance improvements, and enhanced security by encapsulating and managing database logic.

1. Explain the advantages of using stored procedures.

* Stored procedures offer several advantages, including improved performance, enhanced security, code reusability, and reduced network traffic.
* They streamline database tasks by encapsulating logic and enabling efficient, centralized management of business rules.

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

* In SQL, a view is a virtual table that presents data from one or more base tables, providing a simplified or customized perspective on the data. It is essentially a saved query that acts like a table but does not store data itself.
* It dynamically retrieves data from the base tables when accessed. A table, on the other hand, is a physical storage object within the database that holds actual data.

1. Explain the advantages of using views in SQL databases.

* Views are generally used to focus, simplify, and customize the perception each user has of the database. Views can be used as security mechanisms by letting users access data through the view, without granting users permissions to directly access the underlying tables of the query

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

* In SQL, a trigger is a special kind of stored procedure that automatically executes in response to specific events on a table or view. These events typically involve data manipulation language (DML) actions like INSERT, UPDATE, or DELETE.
* Triggers are used to enforce data integrity, automate tasks, and implement business rules.

1. Explain the difference between INSERT, UPDATE, and DELETE triggers.

* INSERT, UPDATE, and DELETE triggers are special stored procedures that automatically execute in response to DML (Data Manipulation Language) events on a table. They differ in the type of event they respond to: INSERT triggers handle new row insertions, UPDATE triggers handle row updates, and DELETE triggers handle row deletions.
* These triggers can be defined as BEFORE triggers, which execute before the DML statement, or AFTER triggers, which execute after the DML statement.

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

* PL/SQL combines the data-manipulating power of SQL with the processing power of procedural languages. You can control program flow with statements like IF and LOOP. As with other procedural programming languages, you can declare variables, define procedures and functions, and trap runtime errors.
* The IF-THEN structure enables conditional execution, while LOOP structures (including FOR, WHILE, and basic LOOP) facilitate repetitive execution.
* IF-THEN Control Structure:
* IF-THEN: Executes a block of statements only if a specified condition is true.
* Syntax:

Code

IF condition THEN

-- Statements to be executed if the condition is true

END IF;

* IF-THEN-ELSE: Executes one block of statements if the condition is true and another if it is false.
* Syntax:

Code

IF condition THEN

-- Statements to be executed if the condition is true

ELSE

-- Statements to be executed if the condition is false

END IF;

IF-THEN-ELSIF: Allows for testing multiple conditions in sequence.

* Syntax:
* Code

IF condition\_1 THEN

-- Statements to be executed if condition\_1 is true

ELSIF condition\_2 THEN

-- Statements to be executed if condition\_1 is false and condition\_2 is true

ELSE

-- Statements to be executed if all conditions are false

END IF;

* LOOP Control Structures:
* LOOP (Basic Loop): Executes a sequence of statements repeatedly until an EXIT statement is encountered.
* Syntax:
* Code

LOOP

-- Statements to be executed repeatedly

-- EXIT; -- Optional: To exit the loop

END LOOP;

* FOR LOOP: Executes a sequence of statements a predefined number of times, typically iterating over a range of numbers or a cursor result set.
* Syntax (numeric range):
* Code

FOR index IN lower\_bound..upper\_bound LOOP

-- Statements to be executed for each value of index

END LOOP;

* Syntax (cursor).
* Code

FOR record\_variable IN (SELECT ... FROM ...) LOOP

-- Statements to be executed for each row returned by the query

END LOOP;

* WHILE LOOP: Executes a sequence of statements repeatedly as long as a specified condition remains true.
* Syntax:
* Code

WHILE condition LOOP

-- Statements to be executed while the condition is true

       END LOOP;

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

* In PL/SQL, a cursor is a control structure that allows you to process individual rows returned by a SQL query. Cursors are used to navigate and manipulate data in a result set, allowing you to access data row by row rather than loading the entire result set into memory at once. There are two types of cursors: implicit and explicit.
* Implicit and explicit cursors in PL/SQL handle SQL statements differently. Implicit cursors are automatically managed by PL/SQL for single-row operations, while explicit cursors require the programmer to manage their lifecycle (open, fetch, close) for multiple-row results.

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

* A SAVEPOINT in transaction management acts as a bookmark within a transaction, allowing you to roll back changes to a specific point without aborting the entire transaction.
* ROLLBACK can be used to revert the transaction to a savepoint, undoing only the changes made after that savepoint, and then the transaction can continue with further operations or be committed. COMMIT finalizes the transaction, making all changes permanent and discarding any savepoints.

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

* SAVEPOINT is used within a transaction to create a named point that can be rolled back to if needed. It is particularly useful for managing complex transactions that may require undoing specific parts without rolling back the entire transaction.