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

➔ SQL (Structured Query Language) is a standardized programming language used to manage and manipulate relational databases.

Data Querying: SQL allows users to query databases to retrieve specific data using commands like SELECT. This makes it easy to access and work with large sets of structured data.

Data Manipulation: SQL provides commands like INSERT, UPDATE, and DELETE to manipulate data. Users can add, modify, or remove records in a database efficiently.

Database Structure Management: SQL includes commands like CREATE, ALTER, and DROP to define and modify the structure of databases and tables. This is crucial for organizing and maintaining data storage.

Data Integrity and Security: SQL supports constraints (such as primary keys, foreign keys, and unique constraints) that ensure data accuracy and consistency. It also allows for user permissions, which helps secure sensitive data.

Data Relationships: SQL enables defining relationships between tables through the use of keys, making it easy to represent complex data models.

Scalability and Performance: SQL can handle large volumes of data and queries efficiently. By using indexing and optimization techniques, it supports scalable database management.

2. Explain the difference between DBMS and RDBMS.

➔

RDBMS	DBMS
Data stored is in table format	Data stored is in the file format

Data in the form of a table are linked together	No connection between data
Support distributed database	No support for distributed database
Data is stored in a large amount	Data stored is a small quantity
RDBMS supports multiple users	DBMS supports a single user
The software and hardware requirements are higher	The software and hardware requirements are low
Example: Oracle, SQL Server.	Example: XML, Microsoft Access.

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

→SQL plays a central role in managing relational databases by providing the necessary tools and commands to interact with, manipulate, and maintain data in an organized, structured way.

1. **Defining Database Structures (Schema Management)**
2. **Inserting, Updating, and Deleting Data (Data Manipulation)**
3. **Querying Data (Data Retrieval)**
4. **Ensuring Data Integrity and Consistency**
5. **Managing Transactions**
6. **Managing Access Control and Security**
7. **Optimizing Database Performance**
8. **Backup and Restoration**

4. What are the key features of SQL?

→

- → Data Querying
- → Data Manipulation
- → Data Definition

- ➔ Data Integrity and Constraints
- ➔ Joins and Relationships
- ➔ Grouping and Aggregation
- ➔ Views
- ➔ Stored Procedures and Functions

1. What are the basic components of SQL syntax?

➔ **KEY WORD:**SELECT, INSERT, UPDATE, DELETE, FROM, WHERE, GROUP BY, ORDER BY, and JOIN

Clauses: SELECT , FROM , WHERE, ORDER BY.

Expressions: AGE>18 , PRICE *5.

Operators: **Comparison operators**: =, !=, <, >, <=, >=.

Logical operators: AND, OR, NOT.

Arithmetic operators: +, -, *, /, %.

Data Types: INT, VARCHAR, DATE, DECIMAL, BOOLEAN.

Comments: Single-line comments start with -- or #, and multi-line comments are enclosed in /*...*/.

Identifiers: employees, salary, department_id.

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

```
➔ SELECT * FROM TABLE_NAME;
        //      ALL RECORD FOR TABLE DISPALY

SELECT ID,NAME FROM TABLE_NAME;
        //      TABLE FOR ALL RECORD CHOICE USER RECORD DISPLAY
```

3. Explain the role of clauses in SQL statements.

→

CLAUSES : SELECE , FROM , WHERE , GROUP BY , HAVING , ORDER BY , LIMIT.

SELECT : RETURN A RESULT SET OF ROW FROM TABLE
FROM : RETURN A RUSULT RETRIVE DATA
WHERE : FILTER DATA ON CONDITION
GROUP BY : GROUP OF DATA IN COLUM
HAVING : FILTER GROUP DATA ON CONDITION
ORDER BY : DATA ASSENDING AND DESENDING
LIMIT : TABLE FOR LIMITED NUMBER FOR ROW

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

→**Constraints** : constraints is used to specify the rule of data table. multiple constraints in sql.

Not null: table value is not emty.
Unique: table value is not dublicate.
Primary key : table value is not emty and dublicate.
Foreign key: two table are link and garneted relationship
Unique auto increment : table are automatic value garneted
Check : used to limit the value range.
Default :used to default value when not insert value.

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

→ The primary key is not null and unique identifier within the table and foreign key is reference one table to primary key to another.

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

   → **Not null:** Role of not null constraint create table and insert the record a not null value in table.

   Unique : role of unique constraint create table and insert the record a unique value in table.

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

→ DDL quary effect of table structure in sql.

Describe DDL create, alter, drop,truncate database object.

Create : create table table name();

Alter : alter table table name add column column name datatype;

alter table table name drop column column name;

alter table table name modify column column name datatype;

drop: drop table table name;

truncate : truncate table table name;

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

→ Create command used database, table, view create in sql.

Create database: Crete database database name;

Create table : create table table name(

```
Id int ,
Name text ,
Salary int
);
```

Create view : create view view name as column name from table name condition;

Select * from view name;

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

→ **Data type:**

Int: declare the positive value.
Float: declare decimal value.
Varchar: declare string value and declare size.
Text: declare string value and not declare size.
Date: declare date atomatically YYYY-MM-DD.
Datetime: declare date and time YYY-MM-DD and hh-mm-ss.
Time: declare time hh-mm-ss.

Constraints:
Not null: table value is not emty.
Unique: table value is not dublicate.
Primary key : table value is not emty and dublicate.
Foreign key: two table are link and garneted relationship
Unique auto increment : table are automatic value garneted
Check : used to limit the value range.
Default :used to default value when not insert value.

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

→ Alter table command in sql use add , drop , modify column name within table.

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

→ **Add column** : alter table tablename add column columnname datatype.

Drop column: alter table tablename drop column columnname.

Modify column: alter table tablename modify column columnname datatype.

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

→ Drop command is use drop (delete) table.

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

→ **Drop table**: Drop table table_name.

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

→ **Insert:** insert into tablename values .

Update: update tablename set condition where condition.
Delete: delete from tablename where condition.

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

→ Update and delete operation execute a specify the condition use where clause importance.

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

→ Select stetment is return a result set of row from table.

Select * from table_name where condition;

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

→ Order by is use multiple data are assending or desendinding order.

Select * from tablename where codition order by columnname.

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

→ Grant and revoke command is type of DCL. grant command is use permitting the user and revoke command is use removing the user.

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

→

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

→ Commit command purpose in sql a save change parmenatly in table, and rollback command purpose in sql a one time undo effect in table.

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

→

Transaction managed in sql database:
Begin transaction : start a transaction.
Commit : save change parmenatly in table.
Rollback : one time undo effect in table.
Save point : save point within transaction are rollback.

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

→ The concept of join in sql to join the table one to more in database.

And ganret the relationship two or more table.
Inner join : return record that matching value in both table.
Left join : return all record that left table and matching value in right table.
Right join : return all record the right table and matching value in left table.
Full join : return all record in both table.

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

→

Inner join :
select tablename.columnname from tablename inner join table name on tablename.columnname= tablename.columnname;

Left join :
select tablename.columnname from tablename inner join table name on tablename.columnname= tablename.columnname;

Right join :

select tablename.columnname from tablename inner join table name on tablename.columnname= tablename.columnname;

Full join :
select tablename.columnname from tablename inner join table name on tablename.columnname= tablename.columnname;

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

→ **GROUP BY Clause in SQL :**

The **GROUP BY** clause in SQL is used to arrange identical data into groups. It is often used with aggregate functions to perform calculations on each group of rows, rather than on the entire result set. The GROUP BY clause groups rows that have the same values in specified columns into summary rows, like summing values, counting records, finding averages .

It is used with aggregate functions :

SELECT column1, aggregate_function(column2) FROM table_name WHERE condition GROUP BY column1;

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

→

Feature	GROUP BY	ORDER BY
Purpose	Group rows based on column values and perform aggregation.	Sort the result set based on column values.
Works With	Aggregate functions (e.g., COUNT(), SUM(), AVG())	Sorting based on column(s) values.
Impact on Results	Reduces the result set by grouping rows.	Does not reduce rows, only sorts them.
Used Before or After Sorting	Before sorting (can be followed by ORDER BY)	After grouping or selecting data.
Common Use	To summarize data (e.g., count, sum).	To sort data (e.g., by date, name, count).

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

→

Procedure : Procedure is like a function but it will never return any value. It will always performed by parameter(argument) or without parameter(argument).

Procedure is used to execute block of code to perform.

- ➔ Procedure without parameter
- ➔ Procedure with parameter

Diffrence :

Feature	Stored Procedure	Standard SQL Query
Definition	A precompiled set of SQL statements stored in the database.	A single SQL statement executed directly.

Execution	Can be executed by calling the procedure name.	Executed as a standalone SQL command.
Reusability	Can be reused multiple times with or without parameters.	Typically written and executed once per use case.
Control Structures	Can contain loops, conditionals, and error handling.	No control flow or error handling (except for WHERE conditions).
Parameters	Can accept input parameters and return output values.	Does not accept parameters (except for bound variables in queries).
Performance	May improve performance due to precompilation and reuse.	Each query is parsed, compiled, and executed separately.
Complexity	Can encapsulate complex business logic and multiple operations.	Generally used for simple, one-time tasks.
Return Values	Can return multiple result sets, output parameters, or status codes.	Returns a result set or a single value (e.g., COUNT()).

2. Explain the advantages of using stored procedures.

→

1. **Improved Performance**
2. **Code Reusability**
3. **Security**
4. **Maintainability**
5. **Consistency and Data Integrity**
6. **Reduced Complexity in Applications**

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

→ A **view** in SQL is a **virtual table** that is based on the result of a **SELECT query**. It is essentially a stored query that can be used like a table in other SQL operations. A view does not store data itself but dynamically retrieves data from one or more underlying tables when queried. It simplifies complex queries by encapsulating them in a reusable object.

Different :

Feature	**View**	**Table**
Data Storage	Does not store data; it's a virtual table.	Physically stores data in the database.
Definition	A stored SQL query that dynamically retrieves data.	A database object that stores data permanently.
Modification	Cannot store data directly; can be updatable or read-only depending on complexity.	Data can be added, updated, and deleted.
Usage	Simplifies querying complex data or joins.	Stores and organizes actual data in rows and columns.
Performance	Slightly slower for complex queries because it re-executes the underlying query each time.	Faster, as it directly stores data, but depends on the query.
Security	Can restrict access to specific columns or rows.	All data in a table is accessible unless restricted by permissions.
Example	CREATE VIEW department_avg_salary AS SELECT department, AVG(salary) FROM employees GROUP BY department;	CREATE TABLE employees (employee_id INT, first_name VARCHAR(50), ...);

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

→

1. Simplification of Complex Queries

2. Security and Access Control

3. Data Abstraction

4. Consistency in Reporting

5. Reusability and Maintainability

6. Data Aggregation

7. Support for Multiple Tables

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

→ A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event.

DML events are INSERT, UPDATE, or DELETE statements on a table or view.

Types of Triggers in SQL

1. DML Triggers (Data Manipulation Language Triggers)

These triggers are fired when data in a table is modified by an **INSERT**, **UPDATE**, or **DELETE** operation. They are the most common types of triggers.

- **INSERT Trigger**: Fired when a new record is inserted into a table.
- **UPDATE Trigger**: Fired when a record is updated in a table.
- **DELETE Trigger**: Fired when a record is deleted from a table.

2. DDL Triggers (Data Definition Language Triggers)

DDL triggers are fired in response to changes in the structure of the database itself, such as **CREATE**, **ALTER**, or **DROP** statements.

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

→

Feature	**INSERT Trigger**	**UPDATE Trigger**	**DELETE Trigger**
Trigger Condition	Fired when a new row is inserted into the table.	Fired when an existing row is updated in the table.	Fired when a row is deleted from the table.
Common Use Cases	- Data validation - Populating default values - Auditing insertions	- Data validation - Enforcing business rules - Auditing updates	- Preventing deletions - Cascading deletes - Auditing deletions
Impact on Data	Affects new data being added.	Affects existing data being modified.	Affects data being removed.
Typical Execution Time	Triggered **after** the INSERT operation.	Triggered **before** or **after** the UPDATE operation.	Triggered **before** or **after** the DELETE operation.
Example Scenario	Track new employee additions.	Update last modified timestamp when data changes.	Prevent deletion of a customer with active orders.

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 (Structured Query Language). It is a **programming language** that enables the execution of complex operations involving SQL queries and control structures within an Oracle database. PL/SQL combines the powerful querying capabilities of SQL with procedural programming constructs like loops, conditionals, variables, and exception handling.

Capabilities :

- ➔ Control Flow
- ➔ Variables and Data Types
- ➔ Error Handling
- ➔ Stored Procedures and Functions
- ➔ Triggers
- ➔ Cursors
- ➔ Batch Processing

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

➔

1. **Reusability**
2. **Modularity**
3. **Error Handling**
4. **Security**
5. **Improved Performance**

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

➔ Control structures in **PL/SQL** are constructs that allow you to control the flow of execution in a PL/SQL block based on conditions or repetitions. These structures are essential for creating dynamic and flexible programs. They include **conditional control structures** (like IF-THEN), **loop control structures** (like LOOP), and **branching mechanisms** (such as EXIT, GOTO, etc.).

1. IF-THEN Control Structure :

The IF-THEN control structure in PL/SQL is used to **make decisions** based on a specified condition. If the condition evaluates to TRUE, the statements within the THEN block are executed. If the condition evaluates to FALSE, no action is taken

Syntax of IF-THEN:

```
IF condition THEN
  -- statements to execute if the condition is TRUE
END IF;
```

2. LOOP Control Structure :

The LOOP control structure in PL/SQL is used to **repeat a block of statements** multiple times. PL/SQL provides several types of loops, including **basic loops**, **FOR loops**, and **WHILE loops**.

Basic LOOP:

A basic LOOP is an infinite loop that will keep executing until a condition is met, often terminated by an EXIT statement.

Syntax:

```
LOOP
  -- statements to be executed repeatedly
  EXIT WHEN condition;  -- exit condition
END LOOP;
```

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

→ Control structures in **PL/SQL** significantly enhance the ability to write **complex queries** and business logic in a relational database environment. By combining **SQL**'s declarative capabilities with **PL/SQL's procedural features**, you can create advanced and dynamic database applications that go beyond simple query execution. Control structures in PL/SQL (like **IF-THEN**, **LOOP**, **CASE**, etc.) allow you to incorporate conditional logic, iterations, and decision-making, which makes it easier to:

- Handle different scenarios or conditions.
- Automate repetitive tasks.
- Implement business rules and complex logic that cannot be accomplished by SQL alone.

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

→A **cursor** in PL/SQL is a **pointer** that refers to a **result set** generated by a query. It allows PL/SQL to handle multiple rows of data that are returned by a query. A cursor is used to retrieve, process, and manipulate the results of a SQL query in a row-by-row manner. In PL/SQL, cursors are especially useful when dealing with **multi-row queries** or performing operations on each row individually.

Types of Cursors

PL/SQL supports two types of cursors:

1. **Implicit Cursors**
2. **Explicit Cursors**

Feature	**Implicit Cursor**	**Explicit Cursor**
Definition	Automatically created by Oracle for single SQL statements.	Declared and explicitly controlled by the programmer.
Control	No direct control over the cursor; Oracle handles everything.	Programmer has full control over the cursor (open, fetch, close).
Usage	Used for single SQL statements like SELECT INTO, INSERT, UPDATE, DELETE.	Used for queries that return multiple rows, and when more control is needed.

Cursor Management	Managed automatically by Oracle.	Must be managed manually by the programmer (open, fetch, close).
Performance	Suitable for simple, single-row queries.	Suitable for complex, multi-row queries that require more processing.
Flexibility	Limited flexibility, as it's designed for simpler queries.	More flexible for complex logic, multi-row processing, and dynamic queries.
Error Handling	Errors related to implicit cursors are less visible.	Errors can be more easily caught and managed during cursor operations.

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

→

Multiple Rows: When your query returns multiple rows, and you need to process each row individually.
Complex Logic: When you need to apply complex logic to each row in the result set (e.g., calculations, updates).
Dynamic SQL: When working with dynamic queries, where the SQL query or filter conditions change at runtime.
Cursor Parameters: When you need to pass parameters to the cursor and filter data dynamically.
Cursor Attributes: When you need to track the cursor's state using attributes like %ROWCOUNT, %FOUND, or %NOTFOUND.

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

→ A **SAVEPOINT** in transaction management is a **marker** or **checkpoint** within a database transaction that allows you to **roll back** part of the transaction to a specific point, without affecting the entire transaction. This provides a mechanism for **partial rollbacks**, giving the user more control over which operations to undo within a transaction.

1.**ROLLBACK to SAVEPOINT**

- If a **ROLLBACK** is issued to a **SAVEPOINT**, the database will undo all changes made after that savepoint, but the transaction will **not be fully rolled back**. Only the operations after the savepoint are undone, and the earlier operations (before the savepoint) remain intact.

Syntax of ROLLBACK to SAVEPOINT:

ROLLBACK TO SAVEPOINT savepoint_name;

2. **COMMIT after a SAVEPOINT**

- When a **COMMIT** is issued, it **commits** all the changes made in the transaction, including any changes that have occurred before and after a savepoint. Once the transaction is committed, all changes become permanent, and no further rollbacks are possible (the transaction ends).

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

→

Complex transactions with multiple steps: Savepoints allow partial rollbacks of specific steps in a long sequence of operations.

Error handling: You can rollback to a savepoint when encountering issues in later steps of a transaction, preserving earlier successful changes.

Testing and simulating: Savepoints allow you to simulate a series of operations and easily undo them if needed.

Conditional operations: When performing conditional logic in a transaction, savepoints allow you to undo only the failed or unnecessary operations without affecting the entire transaction.

Debugging: Helps track where things went wrong and allows you to restore the transaction to a known good state.