**Module 4 Theory Exercise**

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

Ans: Structured Query Language (SQL) is a programming language for storing and processing information in a relational database.

Database management is essential because it ensures data accuracy, consistency, and security, while also improving efficiency and enabling better decision-making.

Why it’s essential:

* + Data integrity and consistency
  + Data security
  + Efficiency and scalability
  + Data management
  + Business intelligence and decision making
  + Regulatory compliance

Q.2 explain the difference between DBMS and RDBMS

Ans:

|  |  |
| --- | --- |
| **DBMS** | **RDBMS** |
| DBMS stores data as a file. | RDBMS stores a data in tabular form. |
| DBMS does not support distributed database. | RDBMS support distributed database. |
| It stores data in either navigational or hierarchical form. | It uses tabular structure where the headers are the column names, and the row contains corresponding values. |
| It deals with small quantity of data. | It deals with large amount of data. |
| It support single user. | It supports multiple user. |

Q.3 Describe the role of SQL in managing relational database.

Ans: SQL plays a crucial role in managing relational databases by providing a standardized to interact with and manipulate data stored within them. It enables users to define database structures, retrieve data, update records, and manage access permissions. Essentially, SQL acts as the bridge between users and the relational database, allowing for efficient and organized data management.

* + Data definition
  + Data retrieval
  + Data manipulation
  + Access control
  + Data integrity
  + Database optimization

Q.4 what are the key features of SQL.

Ans:

1. Data Definition Language (DDL):

SQL provides set of commands to define and modify the structure of database, including creating tables, modifying table structure, and dropping tables.

1. Data Manipulation Language (DML):

SQL provides a set of commands to manipulate data within a database, including adding, modifying, and deleting data.

1. Query Language:

SQL provides a rich set of commands for querying a database to retrieve data, including the ability to filter, sort, group, and join data from multiple tables.

1. Transaction Control:

SQL supports transaction processing, which allows users to group a set of database operations into a single transaction that can be rolled back in case of failure.

1. Data Integrity:

SQL includes features of enforce data integrity, such as the ability to specify constraints on the values that can be inserted or updated in a table, and to enforce referential integrity between tables.

1. User Access Control:

SQL provides mechanism to control user access to database, including the ability to grant and revoke privileges to perform certain operations on the database.

1. Portability:

SQL is a standardized language, meaning that SQL code written for one database management system can be used on another system with minimal modification.

Q.5 what are the basic component of SQL syntax?

Ans: The basic component of SQL syntax is given below:

1. **Data Definition Language (DDL)**

* DDL commands are used to define and manage the structure of database objects like tables, indexes, and views.
* Key DDL commands includes:

CREATE

ALTER

DROP

TRUNCATE

COMMENT

RENAME

1. **Data Manipulation Language (DML)**

* DML commands are used to manipulate data within the database, such as inserting, updating, or deleting records.
* Key DML command includes:

SELECT

INSERT

UPDATE

DELETE

MERGE

CALL

1. **Data Control Language (DCL)**

* DCL commands are used to control access and permissions to the database.
* Key DCL command includes:

GRANT

REVOKE

1. **Data Query Language (DQL)**

* While not always explicitly separated, DQL is often considered a subset of DML and focuses on querying data.
* SELECT

1. **Transaction Control Language (TCL)**

* TCL commands are used to manage transactions, ensuring data consistency.
* Key TCL commands include:

COMMIT

ROLLBACK

SAVEPOINT

SET TRANSACTION

Q.6 write a general structure of an SQL SELECT statement.

Ans: The general structure of an SELECT Statement is: SELECT column\_list FROM table\_name WHERE condition; This statement retrieves data from a specified table based on the provided conditions.

Q.7 Explain the role of clauses in SQL statements.

Ans: Clauses in SQL is a built-in function that is used to retrieve the data from the records present in the database. Different clauses in SQL are used to fetch or retrieve the records from the database table. The ORDER BY clause in SQL is used to arrange the retrieved results in ascending order or descending order.

Q.8 what are constraints in SQL? List and explain the different types of constraints.

Ans: In SQL, constraints are rules applied to database tables to enforce data integrity and maintain data consistency.

Different types of SQL constraints

1. NOT NULL:

Purpose: Ensure that a column contain a NULL value.

Example: A NOT NULL constraints on an email column would mean that every record in the table must have a valid email address.

1. UNIQUE:

Purpose: Ensures that all values in a column are unique, preventing duplicate entries.

Example: A UNIQUE constraints on a username column would ensure that each user has a district username.

1. PRIMARY KEY:

Purpose: Uniquely identifies each record in a table. A table can have only one primary key.

Example: An employee\_id column could be the primary key in an employess table, ensuring each employee has a unique ID.

1. FORIGN KEY:
2. CHECK:
3. DEFAULT:

Q.9 How do PRIMARY KEY and FOREIGN KEY constraints differ?

Ans: \_\_\_\_\_\_\_\_\_\_\_\_

Q.10 what is the role of NOT NULL and UNIQUE constraint?

Ans: 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.

Q.11 Define the SQL Data Definition Language (DDL).

Ans: DDL is a standardized language with commands to define the storage groups, different structures and objects in a database. DDL statements create, modify and remove database objects, such as tables, indexes and storage groups. DDL is also used in a generic sense to refer to any language that describes data.

DDL includes Structured Query Language (SQL) statements to create and drop databases, aliases, locations, indexes, tables and sequences. It also includes statements to alter these objects and impose or drop certain constraints on tables, such as the following:

* UNIQUE
* PRIMARY
* FOREIGN KEY
* CHECK

Q.12 Explain the CREATE command and its syntax.

Ans: The CREATE command in SQL is a Data Definition Language (DDL) command used to create new database objects such as databases, tables, indexes, views, and more. The syntax for the CREATE command varies depending on the object being created.

CREATE DATABASE:

CREATE DATABASE database\_name;

CREATE TABLE:

CREATE TABLE table\_name (

Column 1 data type constraints,

Column 2 data type constraints,

.................

Column N data type constraints,

);

Q.13 What is the purpose of specifying data types and constraints during table creation?

Ans: Specifying data types and constraints during table creation is crucial for ensuring data integrity, accuracy, and reliability within a database.

**Data Types:**

**Purpose:**

Data types specify the format and characteristics of the data that can be stored in a column. This includes things like the type of number (integer, decimal), text, date, or boolean values.

**Constraints:**

**Purpose:**

Constraints are rules that limit the data that can be entered into a table. They enforce data integrity and consistency.

Q.14 What is the use of the ALTER command in SQL?

Ans: The SQL ALTER TABLE statement is used to modify the structure of an existing table in a relational database. It allows you to add, modify, or delete columns, as well as apply constraints, such as primary keys and foreign keys, and make other structural changes to the table.

Q.15 How can you add, modify, and drop columns from a table using ALTER?

Ans: Adding a New Column to a Table. To add a new column to an existing table, you can use the ADD clause. ...

Removing a Column from a Table. We can remove an existing column from a table using the DROP COLUMN clause. ...

Modifying an Existing Column's Data Type.

Q.16 What is the function of the DROP command in SQL?

Ans: In SQL, the DROP command is used to delete a database, table, index, or view. To use the DROP command, you must be very careful because it permanently deletes the object from the database. Therefore, it's essential to backup the object before using the DROP command.

Q.17 What are the implications of dropping a table from a database?

Ans: Dropping a table in a database has several implications, primarily involving the permanent removal of the table's structure and data, along with its associated dependencies. This action is irreversible and can impact other database objects and user access.

Here's a breakdown of the implications:

* Data Loss
* Object Removal
* Dependency Removal
* Authorization and Access Changes
* Space Reclamation
* Invalidation of Dependent Objects
* Potential Errors and Inconsistencies

Q.18 Define the INSERT, UPDATE, and DELETE commands in SQL

Ans: The INSERT statement introduces new rows into an existing table.

The DELETE statement removes a row or combination of rows from a table.

The UPDATE statement enables users to update a row or group of rows in a table.

Q.19 What is the importance of the WHERE clause in UPDATE and DELETE operations?

Ans: The DELETE and UPDATE statements should contain a WHERE clause in order to limit the scope of the operation. Without a WHERE clause, the statement will delete or update all rows in a table, which is not usually the desired outcome.

Q.20 What is the SELECT statement, and how is it used to query data?

Ans: The SELECT statement in SQL is used to query and retrieve data from a database. It specifies which columns and rows to retrieve from one or more tables, and can include conditions for filtering and sorting the results.

Purpose:

The primary function of the SELECT statement is to fetch data. It's the foundation for querying information from a database.

Basic Syntax:

SELECT column1, column2, ... specifies the columns to be retrieved.

FROM table\_name indicates the table to retrieve the data from.

Q.21 Explain the use of the ORDER BY and WHERE clauses in SQL queries.

Ans: The WHERE clause in SQL filters data, selecting only rows that meet specific conditions. The ORDER BY clause sorts the resulting data, either in ascending or descending order, based on specified columns.

WHERE Clause:

* Purpose:

Filters rows from a table based on specified conditions.

ORDER BY Clause:

* Purpose:

Sorts the result set of a query.

Q.22 What is the purpose of GRANT and REVOKE in SQL?

Ans:

|  |  |  |
| --- | --- | --- |
| **S.NO** | **GRANT** | **REVOKE** |
| 1 | GRANT command is used to give access privileges to the users or other rights or opportunities for the database. | The REVOKE command does just opposite to the GRANT command. It withdraws user privileges on database objects. |
| 2 | It authorizes access preferences to users. | It withdraws access preferences to users. |
| 3 | In the GRANT command, you need to define the permissions for each user. | In the REVOKE command, if the access for one user is withdrawn, then all the permissions provided by that particular person to others will also be removed. |
| 4 | GRANT is easy to execute. | REVOKE is hard to perform. |

Q.23 How do you manage privileges using these commands?

Ans:

* Granting Privileges
* Revoking Privileges
* Managing Privileges with Roles
* Managing Privileges in Specific Environments

Q.24 What is the purpose of the COMMIT and ROLLBACK commands in SQL?

Ans: 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.

Q.25 What is the purpose of GRANT and REVOKE in SQL?

Ans:

REVOKE: The REVOKE statement can cancel any of the following access privileges or roles that a user, or PUBLIC, or a role currently holds: Privileges on the database (but a role cannot hold database-level privileges) Privileges on a table, synonym, view, or sequence object.

GRANT: Grant Command: This is a SQL command which is used to provide privileges/permissions to modify and retrieve database objects like tables, views, sequences, indexes, and synonyms. This command also gives privileges like providing the same permissions to some third user as well.

Q.26 How do you manage privileges using these commands?

Ans: To manage privileges in databases, the GRANT and REVOKE commands are used. GRANT is used to assign privileges to users or roles, while REVOKE is used to remove those privileges. These commands allow control over who can access specific database objects and what operations they can perform (e.g., SELECT, INSERT, UPDATE, DELETE).

Q.27 What is the purpose of the COMMIT and ROLLBACK commands in SQL?

Ans: 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.

Q.28 Explain how transactions are managed in SQL databases.

Ans: If a transaction is successful, all of the data modifications made during the transaction are committed and become a permanent part of the database. If a transaction encounters errors and must be cancelled or rolled back, then all of the data modifications are erased. Each individual statement is a transaction.

Q.29 Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFTJOIN, RIGHT JOIN, and FULL OUTER JOIN?

Ans: In SQL, a JOIN clause combines rows from two or more tables based on a related column between them. Different types of JOINs offer various ways to handle matching and non-matching records.

|  |  |  |  |
| --- | --- | --- | --- |
| INNER JOIN | LEFT JOIN | RIGHT JOIN | FULL JOIN |
| * Returns only the rows where there is a match in both tables based on the join condition. | Returns all rows from the left table (the first table in the join) and the matching rows from the right table. If there's no match in the right table, NULL values are used for the right table's columns. | * Returns all rows from the right table and the matching rows from the left table. If there's no match in the left table, NULL values are used for the left table's columns. | Returns all rows from both tables. If there's a match, the data is combined. If there's no match, NULL values are used for the missing columns. |
| * If a row in one table doesn't have a corresponding match in the other, it's excluded from the result. | Using the Customers and Orders tables again, a LEFT JOIN would show all customers, even those who haven't placed any orders. For customers without orders, the order-related columns would be NULL. | A RIGHT JOIN on Customers and Orders would return all orders and the customer information for each order, even if the customer information is not available for a particular order. | A FULL OUTER JOIN on Customers and Orders would show all customers and all orders, combining the information where it matches and filling in with NULLs where there's no match on either side. |

Q.30 how are joins used to combine data from multiple tables?

Ans: SQL JOIN. A JOIN clause is used to combine rows from two or more tables, based on a related column between them. Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.

Q.31 What is the GROUP BY clause in SQL? How is it used with aggregate functions?

Ans: The GROUP BY clause in SQL is used to group rows that have the same values in one or more columns into summary rows, like calculating the average salary per department. It is frequently used with aggregate functions (like SUM, AVG, COUNT, MAX, MIN) to perform calculations on these groups. Essentially, GROUP BY divides the result set into groups, and then aggregate functions operate on each group independently.

How it's used with aggregate functions:

**Select columns and aggregate functions:** You select the columns you want to group by and the aggregate functions you want to apply.

**GROUP BY clause:** After the SELECT clause and WHERE clause (if any), you add the GROUP BY clause, specifying the column(s) to group by.

Q.32 Explain the difference between GROUP BY and ORDER BY.

Ans:

|  |  |  |
| --- | --- | --- |
|  | GROUP BY | ORDER BY |
| Purpose | Groups rows that have the same values in specified columns. | Sorts the result set of a query based on one or more columns. |
| Functionality | Used to organize data into groups, allowing for calculations (like sums, averages, counts) on each group using aggregate functions. | Arranges the output in ascending (default) or descending order. |
| Placement | GROUP BY is placed after the WHERE clause (if present) and before the ORDER BY clause. | ORDER BY is placed at the end of the query. |
| Example | You might use GROUP BY to find the total sales per product category or the average score per student. | You might use ORDER BY to display employees by salary (highest to lowest) or products by name (alphabetically). |

Q.33 What is a stored procedure in SQL, and how does it differ from a standard SQL query?

Ans: A stored procedure in SQL is a precompiled set of SQL statements stored within the database, designed for repeated execution and reuse, while a standard SQL query is a single statement executed on demand. Stored procedures offer benefits like improved performance, enhanced security, and code modularity, whereas standard queries are simpler for one-off data retrieval or manipulation.

Q.34 Explain the advantages of using stored procedures

Ans: Advantages of Stored Procedures:

* **Improved Performance:**

Stored procedures are compiled once and stored in an executable form within the database, leading to faster execution compared to sending multiple individual queries. The compiled code is also cached, further reducing execution time and overhead.

* **Reduced Network Traffic:**

By grouping SQL statements into a single procedure, only a call to the procedure needs to be sent over the network, rather than individual queries, reducing network load and improving response times.

* **Enhanced Security:**

Stored procedures can be used to control access to data, granting users permission to execute specific procedures without needing direct access to the underlying tables. This helps prevent unauthorized data access and potential security breaches like SQL injection attacks.

* **Code Reusability and Maintainability:**

Stored procedures encapsulate business logic, allowing developers to reuse the same code across multiple applications and systems, reducing redundancy and simplifying maintenance. Modifications to the logic only need to be made in one place, the stored procedure, rather than in multiple applications.

* **Centralized Management:**

Stored procedures promote centralized management of database logic, making it easier to update and maintain business rules across different applications.

* **Simplified Development:**

Stored procedures can encapsulate complex queries and operations, making it easier for developers to build applications without needing to write lengthy and complex SQL queries directly within the application code.

* **Better Error Handling:**

Stored procedures provide a structured way to handle errors using features like TRY...CATCH blocks, improving the robustness of database interactions.

Q.35 What is a view in SQL, and how is it different from a table?

Ans: In SQL, a view is a virtual table based on the result of a query, while a table is a physical storage structure that holds data. Essentially, a view doesn't store data itself; it retrieves and presents data from one or more base tables according to a stored query.

Q.36 Explain the advantages of using views in SQL databases

Ans:

1. Simplification of Complex Queries:

* Views can encapsulate complex joins and aggregations, presenting a simplified, single-table view of the data.
* This makes it easier for users to query the data without needing to understand the underlying complexities of the database schema.
* For example, a view could combine data from multiple tables into a single, easily queryable view, simplifying multi-table queries.

1. Enhanced Data Security:

* Views can restrict access to specific columns or rows within a table, providing a layer of data security.
* Users can be granted access to a view without having direct access to the underlying tables, limiting their exposure to sensitive data.
* This is particularly useful in multi-user environments where different users require access to different subsets of data.

1. Data Abstraction and Flexibility:

* Views provide a level of abstraction, hiding the underlying complexity of the database structure from users and applications.
* This allows for changes to the underlying tables without requiring modifications to applications that rely on the views.
* For instance, if a table is restructured or renamed, the view can present a consistent representation of the data, even if the underlying structure changes.

1. Code Reusability:

* Views can be reused across multiple queries and applications, reducing code duplication and promoting consistency.
* Instead of writing the same complex query multiple times, a view can be created once and then referenced by different queries.
* This makes code maintenance easier and more efficient.

1. Improved Performance with Materialized Views (in some cases):

* While standard views don't store data, materialized views pre-compute and store the results of a query.
* This can significantly improve query performance, especially for complex queries that are executed frequently.
* However, materialized views require storage space and need to be refreshed when the underlying data changes.

Q.37 What is a trigger in SQL? Describe its types and when they are used

Ans: In SQL, a trigger is a special kind of stored procedure that automatically executes in response to a specific event (like an INSERT, UPDATE, or DELETE) on a table or view. They are used to enforce business rules, maintain data integrity, and automate tasks. Triggers can be categorized by when they execute (BEFORE or AFTER) and the type of event that triggers them (INSERT, UPDATE, or DELETE).

* **When To use Trigger:**

1. **Enforcing Business Rules**
2. **Maintaining Data Integrity**
3. **Auditing and Logging**
4. **Automating Tasks**

Q.38 Explain the difference between INSERT, UPDATE, and DELETE triggers.

Ans:

|  |  |  |  |
| --- | --- | --- | --- |
|  | INSERT TRIGGER | UPDATE TRIGGER | DELETE TRIGGER |
| Purpose | Fires when a new row is inserted into a table. | Fires when an existing row is modified. | Fires when a row is deleted from a table. |
| Use Cases | Validating data before insertion.  Logging new records for auditing.  Populating default values based on other columns in the newly inserted row. | Auditing changes to specific columns.  Enforcing referential integrity | Logging deleted records for auditing.  Enforcing referential integrity by deleting related records.  Maintaining consistency between tables. |
| Accessing Data | INSERT triggers can access the newly inserted data through the NEW pseudo-table (e.g., NEW.column\_name). There is no OLD pseudo-table in an INSERT trigger because the row doesn't exist before the insertion. | UPDATE triggers can access both the old (pre-update) and new (post-update) values of a row using the OLD and NEW pseudo-tables respectively. | DELETE triggers can access the deleted data through the OLD pseudo-table (e.g., OLD.column\_name). There is no NEW pseudo-table in a DELETE trigger because the row is already removed. |

Q.39 What is PL/SQL, and how does it extend SQL's capabilities?

Ans: PL/SQL is a procedural language designed specifically to embrace SQL statements within its syntax. PL/SQL program units are compiled by the Oracle Database server and stored inside the database. And at run-time, both PL/SQL and SQL run within the same server process, bringing optimal efficiency.

 These extensions include user-defined functions, language extensions (like in SQL Server), and database-specific features that enhance query capabilities.

* User-Defined Functions (UDFs)
* Language Extensions
* Database-Specific Extensions
* Extending SQL for Analytics
* Extending SQL for Development

Q.40 List and explain the benefits of using PL/SQL

Ans:

**1. Tight Integration with SQL:** PL/SQL is designed to work seamlessly with SQL, allowing developers to embed SQL statements directly within PL/SQL code for data manipulation and retrieval. This eliminates the need for separate SQL calls, improving performance and reducing network traffic.

**2. High Performance:** By executing blocks of code at once and reducing network round trips, PL/SQL enhances application performance, especially for complex database operations. It also allows for optimized data processing within the database server, minimizing the overhead of data transfer between the application and the database.

**3. High Productivity:** PL/SQL's procedural language capabilities, coupled with its SQL integration, allow developers to write compact and efficient code for querying, transforming, and updating data. Features like exception handling and object-oriented programming support further contribute to increased productivity by reducing development time and debugging efforts.

**4. Portability:** PL/SQL applications are highly portable across different Oracle database platforms, enabling developers to deploy and run their code on various systems without modification. This reduces the complexity of managing different codebases for different environments.

**5. Security:** PL/SQL offers robust security features, including the ability to encapsulate sensitive operations within stored procedures and functions, and restrict access to authorized users. This enhances the security of the database by preventing unauthorized access and manipulation of data.

**6. Error Handling:** PL/SQL provides comprehensive error handling mechanisms, allowing developers to catch and manage runtime errors gracefully. This ensures that applications continue to function smoothly even when unexpected errors occur, improving the user experience and application stability.

**7. Reusability:** PL/SQL allows developers to create reusable code blocks in the form of stored procedures, functions, and packages. This modular approach promotes code reuse, reduces development time, and ensures consistency across different parts of an application.

**8. Support for Object-Oriented Programming:** PL/SQL supports object-oriented programming concepts, allowing developers to define objects, methods, and encapsulate data within objects. This enables the development of more complex and maintainable applications.

**9. Web Application Development:** PL/SQL can be used to develop web applications by generating dynamic web pages directly from the database. This simplifies the development of web-based applications that interact with the database.

**10. Manageability:** PL/SQL's block structure, modularity, and features like exception handling and data hiding make it easier to manage and maintain complex applications.

Q.41 What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.

Ans: In PL/SQL, control structures dictate the flow of execution in a program. They include conditional statements like IF and looping statements like LOOP. The IF-THEN structure executes a block of code only if a specified condition is true, while the LOOP structure allows repeating a block of code until a certain condition is met or a specified number of iterations are completed.

**IF-THEN Control Structure**

The IF-THEN statement allows execution of a block of code based on a boolean condition. If the condition evaluates to TRUE, the code within the THEN clause is executed. If the condition is FALSE, the code is skipped.

IF condition THEN

-- Code to execute if condition is TRUE

END IF;

DECLARE

v\_number NUMBER := 10;

BEGIN

IF v\_number > 5 THEN

DBMS\_OUTPUT.PUT\_LINE('Number is greater than 5');

END IF;

END;

/

**LOOP Control Structure**

The LOOP statement in PL/SQL provides a way to execute a block of code repeatedly. It requires an END LOOP; statement to mark the end of the loop and an optional EXIT statement to terminate the loop prematurely based on a condition.

LOOP

-- Code to execute repeatedly

-- Optional: EXIT WHEN condition;

END LOOP;

DECLARE

v\_counter NUMBER := 1;

BEGIN

LOOP

DBMS\_OUTPUT.PUT\_LINE('Counter: ' || v\_counter);

v\_counter := v\_counter + 1;

EXIT WHEN v\_counter > 5;

END LOOP;

END;

/

Q.42 How do control structures in PL/SQL help in writing complex queries?

Ans: Control structures enable developers to dictate the flow of execution in a program. They form the core of logic building, allowing decisions, repetitive tasks, and conditional executions to be seamlessly handled.

Q.43 What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.

Ans: In PL/SQL, a cursor is a mechanism for handling the result set of a SELECT statement, allowing row-by-row processing of data. There are two types: implicit and explicit. Implicit cursors are automatically created by PL/SQL for simple SELECT statements, while explicit cursors are declared and managed by the programmer for more complex scenarios.

|  |  |  |
| --- | --- | --- |
| Feature | IMPLICIT CURSOR | EXPLICIT CURSOR |
| Declaration | Automatic | Programmer-defined |
| Control | Limited | Full |
| Rows Returned | Single Row or single row from DML | Multiple Row |
| Use Cases | Simple queries, DML operations | Complex query, multi-row processing |
| Example | INSERT INTO table\_name VALUES (1,’value’) | DECLARE CURSOR c1 IS SELECT \* FROM employees; |

Q.44 When would you use an explicit cursor over an implicit one?

Ans: Implicit cursors are great for the simple tasks while explicit the cursors are better for more complex jobs where you need the precise control over the data. Beginners mostly start with the implicit cursors because they are easier and then learn to use the explicit cursors as they tackle more advanced databases tasks.

s